# Multistage Decisions And Risk In Markov Decision Processes: Towards Effective Approximate Dynamic Programming Architectures

A Thesis
Presented to
The Academic Faculty

by

**Nikolaos E. Pratikakis**

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Chemical and Biomolecular Engineering
Georgia Institute of Technology
December 2008

# Multistage Decisions And Risk In Markov Decision Processes: Towards Effective Approximate Dynamic Programming Architectures

Approved by:

Professor Jay H. Lee, Advisor
School of Chemical and Biomolecular Engineering
*Georgia Institute of Technology*

Professor Shabbir Ahmed
School of Industrial and Systems Engineering
*Georgia Institute of Technology*

Professor Matthew J. Realff, Advisor
School of Chemical and Biomolecular Engineering
*Georgia Institute of Technology*

Professor Martha Grover-Gallivan
School of Chemical and Biomolecular Engineering
*Georgia Institute of Technology*

Professor Stylianos Kavadias,
College of Management
*Georgia Institute of Technology*

Date Approved : 20 August 2008

*To the memory of my grandfather*

*Κοστα*

# ACKNOWLEDGEMENTS

Thank God, I think I am done!

First and foremost, I need to acknowledge my advisors at Georgia Tech (In alphabetical order): Jay H. Lee, and Matthew J. Realff. They complement each other in the most unique way. I thank them for giving me the opportunity to explore and learn these very different areas of science. Looking back, through my countless meetings, it was a unique experience to share or at times not to share new ideas with them! I owe them a lot, especially my development as a responsible professional individual. I thank them for believing in me, pushing me to my limit and trusting me that I will eventually blossom with interesting ideas.

After the advisors, I would like to thank all the individuals and coworkers that help me persevere. Starting from day one I made this trip with a dear friend and colleague Christos Fountoukis. We shared the same experiences for about 3.5 years; from taking courses to having fun together. The time we spent together is always going to be cherished by me. I would like to thank my friend Kevin P. Doyle for being a great roommate for almost 2 years. If he ever reads this he should try to contact me. I also would like to thank my two dear friends Eduardo and Charlene for their encouragement of pursuing my PhD and for helping me adapt to my new environment. I will make sure i will visit them and vice versa, i wish them the best in their personal and professional life. I would like to thank my colleague and dear friend Wee Chin Wong. I have had endless discussions with Wee Chin, he is remarkable and brilliant. I admire his intelligence and insights, he has set a standard of excellence for me. In the future, I look forward collaborating with him at a professional level. I would like to thank Vasileios Papapostolou. Our friendship started at the National Technical University Of Athens and then continued after his transition to Harvard University. He and I definitely need rollover minutes!

I would like to thank the seniors who graduated before me: Anshul a remarkable individual who had answers for almost anything, Niket who had the kindest of heart and time for anybody, Jongmin Lee for being a Korean brother!, Jaein for wanting to teach me his work but never had the time (I do understand now why), Tina for demonstrating what discipline and professionalism really means. I would like to thank Rakshita for the many conversations. I love her attitude towards life and I think I have never seen her stressed. I would also like to mention the other current and former members of the group whose company I have enjoyed over the last few years Fermi,Ugur,Prabudha, Choon Meng and Jihoon, and I advise them to persevere to the end.

I will also like to thank Ms. Janice Whatley and Ms. Juanita Freeman for providing the solution to every administrative problem as a graduate student. I extend my gratitude to the other members of my committee Dr. Martha Grover-Gallivan, Dr Stylianos Kavadias and Dr Shabbir Ahemed for pushing me hard when i did my third-fourth year progress evaluation. I also would like to thank former N.T.U.A Academic advisor Professor Andreas Boudouvis and current Georgia Tech ChBE faculty member Professor Athanasios Nenes for the constant moral support.

At this point, I would also like to mention some people who have affected not only my professional, but also my personal life. Starting with my family in Greece; my parents have been a supporting hand on my shoulders all of my life. They trusted every decision I made and were full of encouragement. My mom for talking to me every single day at least twice. We must of have spent a fortune in calls. My dad for going though two surgeries, working all of his life for my well being and education and providing for me always. My sister for our endless fights and for her patience to listen to my mother's complaints about my absence. There are many other members of my extended family who have helped me at different points in my life and I would like to thank all of them. At the very end, I would like to thank my lovely wife Kimberly Pratikakis and my 2 year old son Manolis. Their companionship certainly made things a lot easier for me (most of the times!). We have a great family. My son is the light in my life. I hope I will instill to him some of the good qualities that were instilled in me in these 5 years. I would like to thank my wife for being a

loving and most beautiful friend, girlfriend, and wife, a man could have asked for. I cherish the last four years (I think it's four, I may be mistaking here!) deeply and look forward to the rest of my life with you.

Last, I will like to say goodbye and goodnight -($\kappa\alpha\lambda\eta\nu\upsilon\chi\tau\alpha$) to my grandfather and I dedicate this thesis to his memory. I didn't attend his funeral because my son was born at the same time. My grandmother told me that when he heard that Manolis was born he smiled...

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# SUMMARY

The scientific domain of this thesis is optimization under uncertainty for discrete event stochastic systems. In particular, this thesis focuses on the practical implementation of the Dynamic Programming (DP) methodology to discrete event stochastic systems. Unfortunately DP in its crude form suffers from three severe computational obstacles that make its implementation to such systems an impossible task. This thesis addresses these obstacles by developing and executing practical Approximate Dynamic Programming (ADP) techniques.

Specifically, for the purposes of this thesis we developed the following ADP techniques. The first one is inspired from the Reinforcement Learning (RL) literature and is termed as Real Time Approximate Dynamic Programming (RTADP). The RTADP algorithm is meant for active learning while operating the stochastic system. The basic idea is that the agent while constantly interacts with the uncertain environment accumulates experience, which enables him to react more optimal in future similar situations. While the second one is an off-line ADP procedure. Both approaches are developed for discrete event stochastic systems and their main focus is the controlled exploration of the state space circumventing in such a way one of the severe computational obstacles of DP that is related with the cardinality of the state space.

These ADP techniques are demonstrated on a variety of discrete event stochastic systems such as: **i)** a three stage queuing manufacturing network with recycle, **ii)** a supply chain of the light aromatics of a typical refinery and **iii)** several stochastic shortest path instances with a single starting and terminal state.

Moreover, this work addresses, in a systematic way, the issue of multistage risk within the DP framework by exploring the usage of single-period and multi-period risk sensitive utility functions. In this thesis we propose a special structure for a single-period utility and compare the derived policies in several multistage instances. Finally, we briefly attempt

to intergrade the developed ADP procedures with the proposed utility to yield ADP risk sensitive policies.

# CHAPTER 1

# INTRODUCTION

For large scale decision making problems, approximate dynamic programming (ADP) [1] has emerged as an effective way to approximate the conceptually elegant but computationally inefficient dynamic programming algorithm [2]. In this thesis, I will focus on developing and applying the ADP technique to problems from Process System Engineering (PSE). I present a classification of PSE problems in Figure 1 and will place my contributions within this context.

## *1.1 Thesis Focus On Multistage Decision Problems Under Uncertainty*

The bottom layer of this decision hierarchy involves regulatory control and such problems are addressed via decentralized control strategies [3]. The control engineer is interesting in controlling the set point of complex processes with Multiple Inputs and Multiple Outputs (MIMO systems). In practise the most common solution is to match one input with a single output in order to achieve the desired set points.

The set points usually change during the plant operation either intentionally to maximize the plant's financial objective or to reject long term disturbances. Solving the MIMO without coordination will have a big impact on the systems performance. Therefore the set points are usually inputs to Model Predictive Control (MPC) formulations, which are numerical rolling horizon mathematical programming control strategies [4]. Significant gains can be achieved via MPC formulations [4]. The tools that this thesis produces can address problems formulated as an MPC, but are more suited for longer decision horizon problems, where explicit uncertainty can be modeled and process dynamics are less of a concern. In general, strategic problems are problems in which the future uncertainty must be taken into consideration for the optimal solution. Examples of such problems are : 1) manufacturing

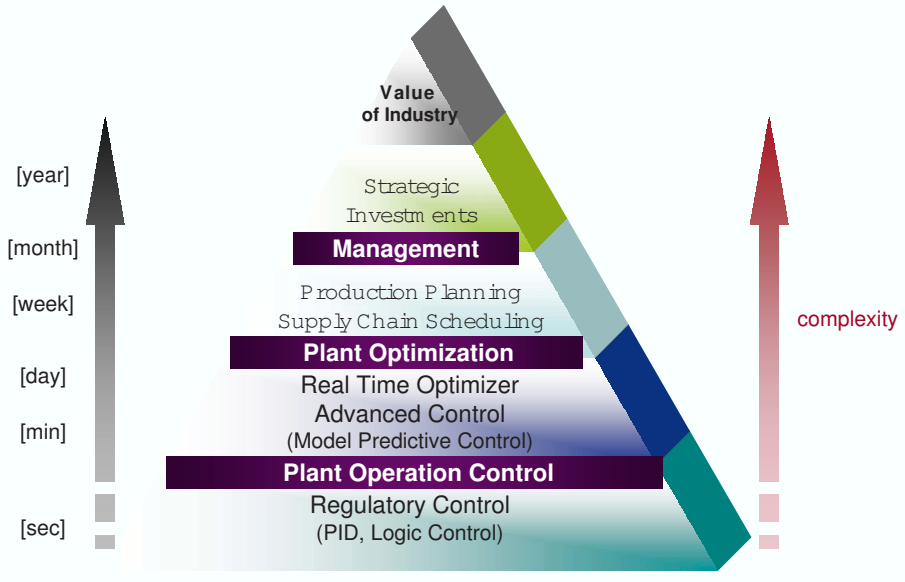



**Figure 1:** Decision Hierarchy In Modern Process System Industries.

capacity planning problems under demand uncertainty (*Elberly and Mieghem*) [5] , 2) supply chain operation and design under demand uncertainty (*Santoso et al.*)[6], 3) project portfolio management problems under stochastic arrival rates and progress/failure rate uncertainty (*Rogers et al.*) [7], 5) optimal pair trading problems (*Mudchanatongsuk et al*) [8], 6) hedge funds (*Primbs*) [9], 7) technology adoption problems (Ulu and Simth) [10].

In these problems the consequences of actions taken today can significantly influence the outcomes in the more medium to long-term future. It is important to capture the multi-stage nature of those problems and the fact that the trajectory of the system can be remarkably different based on decisions taken early on. In the next section, we design a specific shortest path example in order to clarify and underline what is considered a single stage problem versus a multistage one.

To address multistage problems the decision maker must take into consideration of all the possible future trajectories, weigh them with a corresponding probability, and decide accordingly. The number of trajectories-scenarios needed to be examined are exponential to the horizon length and therefore to do this explicitly is computationally infeasible for modest size problems.

Given that the size of these multistage decisions problems will be very large, this motivates the development of heuristic methods to address them. In addition, as the problems transition from tactical planning to the strategic decision-making, the measure and incorporation of risk into the decision making becomes important.

## *1.2 Handling Risk In Single Stage Problems: A Shortest Path Example*

In this section, I consider a one dimensional single stage shortest path problem with a single starting position $(x, y) = (0, 0)$ and a goal position $(x, y) = (10, 6)$ (Fig.2). Via this example I want to describe in a simplistic fashion several important concepts for this thesis. These are: a) the State space, b) the Action space, c) what is regarded as Risk, d) the difference of the One stage (but multi-step optimization) Vs. Multi-stage optimization, e) what is a Policy.

This problem of reaching the goal consists of multiple steps, but has only one stage because you do not need to revise the decisions during the traversal. The uncertainty lies on the cost structure, since one incurs a normally distributed cost $f_{(x,y)} \sim N(\mu_{(x,y)}, \sigma_{(x,y)})$, when a particular $(x, y)$ position is visited. From a systems perspective each position is regarded as the state of the system. Here, from each position the agent faces eight choices, {Up,Down,Left,Right,4 Diagonal moves}. Depending on the field, such choices are often called actions or decision variables (Mathematical Programming field) or controls (Process Control field). All the possible states of the system compose the so-called state space, all the possible choices compose the so-called action space. In general, if we are aware of the state of the system, we would be able to find the best possible action that would maximize/minimize the expected profit/cost.

My goal is to provide a methodology for large discrete event systems that would be able to generate, via optimization, a wide spectrum of policies (in this case paths) for the decision maker. By the term decision policy we refer to a mapping indicating what action to take given the system state. In particular, the derived paths should be risk-sensitive, meaning that I will optimize against a linear combination of two contradictory objectives. The

first objective would be to minimize/maximize the expected cost/profit, while the second objective would worry about how badly I could do, in essence minimizing/maximizing the objective if the worst case scenarios were to be realized.

The qualitative comments about the quantitative results-paths generated for such a one stage but multi step single stage problem as it appears at Fig.2 are summarized as follows:

- Path 1: Minimizing the expected cost while disregarding the second objective. For this instance, the shortest path that minimizes the expected cost is represented with the solid line at Fig.2. The mean cost of that policy is 202 with a standard deviation of 32.5. While the corresponding risk measure if I follow this path is 269.9. The risk measure indicates how badly I can do, in an expected sense for the 5% of the worst cases.

- Path 2: Minimize the risk measure as discussed above, which indicates how badly I could do if I follow a specific path while disregarding the first objective. For this instance, this path is represented with the dashed line at Fig.2. The mean cost is 206 with a standard deviation of 14.5. While the corresponding risk measure that indicates how badly I can do if I follow this path is 235.9. In this case, we did not optimize we respect to the first objective, the fact that the expected cost is close to 202 (mean of path 1) is symptomatic.

- In the case that we are interesting in optimizing both objectives, we would assign a corresponding weight to each of them and perform the optimization. For instance, if we equally weight them we retrieve a policy where the mean cost is 205 with a standard deviation of 21. While the corresponding risk measure that indicates how badly I can do if i follow this path is 248.2.

**Why is this problem a single stage problem?** This problem is a single stage problem, because there is no need to revise any decisions during the execution of the plan. In subsequent chapters the shortest path problem will be used for illustration, but will have a multi-stage structure because the actions will not always lead to the same outcome and the choice whether to go next will depend on what state you end up in.

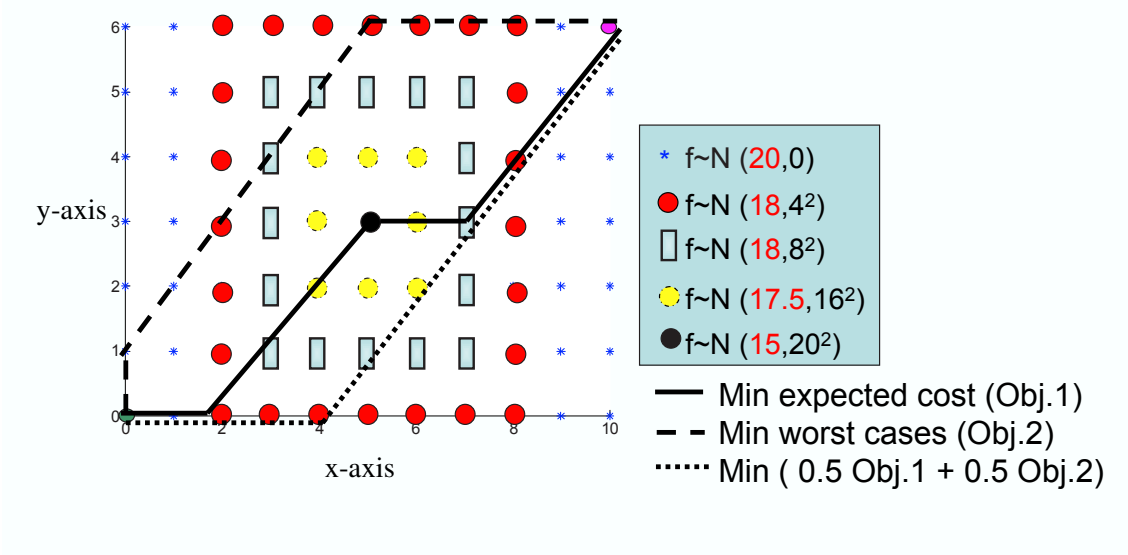



**Figure 2:** Schematic illustration of a shortest path problem. We illustrate, how appropriating weighting on the objectives can yield a different policy.

**Table 1:** The mean $\mu$ and standard deviation $\sigma$ of the cost distributions associated with the different policies according to $\lambda$ values. ($\alpha$=0.95)

| | Expected Cost ± Standard Deviation | How Badly I Will Do Given The Path |
|---|---|---|
| Minimize Expected Cost | 202 ± 32.5 | 269.9 |
| Minimize Expectation How Badly I could Do | 206 ± 14.5 | 235,9 |
| Equally Weight Both Objectives | 205 ± 21 | 248.2 |

The end goal of this work would be to utilize this linear combination of two contradictory objectives in the multi-stage problem within promising ADP methodologies

## *1.3 Thesis Scope And Structure*

This thesis is concerned with general multistage problems under risk neutral and risk-sensitive objectives. For this purpose, I formulate the problems of interest as Markov decision processes (MDP) [11] with a finite number of states and actions, and consider indefinite planning horizons. Some problems will have dedicated goal states present, while some other problems will not. For all the studied problem, we assume that we have a valid model and a Markovian model for its random variables.

The first part of the thesis is focused on creating effective Approximate Dynamic Programming (ADP) methodologies and finding policies that maximize/minimize a risk-neutral objective.

The second part of this thesis develops a method to find policies that are differentiated by sensitive behaviors. To do so we explore the usage of single and multi-period utilities within exact DP and ADP approaches [12, 13, 14].

The structure and the contributions of this thesis are the following:

1. In Chapter 2, I provide some necessary background for this thesis. First, I examine the issues of deterministic optimization when applied to multistage problems, and communicate the fact that following advances in mathematical programming are embraced by the designed Approximate Dynamic Programming (ADP) methodologies of this work. Then, I define the formalism of a Markov Decision Process and the main ideas of DP. Finally, I go through some basics in utility theory and risk measures and review several methodologies addressing the issue of multistage risk.

2. In Chapter 3, I propose several modifications to the Real Time Dynamic Programming (RTDP) algorithm initially proposed by *Barto el al* [15] in order to explore the tradeoff between the exploration of the state space and the exploitation of existing policies. The exploration-exploitation trade off is the classic dilemma in RL-based algorithms (*Suttton*)[16] . Specifically the decision maker will either exploit its existing knowledge executing the greedy control or decide to explore different actions just in case he/she discovers states of the world that will eventually lead the system to achieve a better steady-state performance. The proposed RTDP modifications will target reducing the computational obstacles associated with DP based on its different curses of dimensionality [1].

   In the third chapter I explore these issues in the context of an exemplary capacity planning manufacturing system with inventory flow decisions, which faces uncertainty in demand and intermediate quality, as well as several shortest path instances.

3. In Chapter 4, I formulate a large scale supply chain light aromatics case study as a

MDP and compare the RTADP solution against mixed integer and two-stage stochastic mathematical programming rolling horizon formulations.

The motivation for this study is to quantify the impact of relative timing of decisions and information flow. This will provide to the reader a quantitative picture about the value of receiving information before implementing planning decisions.

This application is ideal for this study since its decision space may be characterized by two different time scales. In practice, there are many industrial settings that share a similar decision space.

4. In Chapter 5, I delineate a different ADP approach that is designed to perform a controlled exploration of the state space. This approach utilizes Monte Carlo simulations and Approximate Value Iteration in an iterative fashion and is regarded by this author as an evolution to the established ADP in a heuristically confined state space originally proposed by *Choi et al* [17, 18].

   The proposed approach performs a controlled exploration of the state space and is particular useful for applications with dedicated goal states. I evaluate its computational behavior with respect to the algorithmic tuning parameters and also demonstrate the practical use of the approach in more complex applications by tailoring the manufacturing capacity planning example that we studied before to fit the description of a shortest path problem. The results derived from this ADP approach are compared against a rolling horizon mathematical programming optimization strategy.

5. In Chapter 6, I shift from risk neutral-decision-making problems to problems where one accounts for risk sensitivity in MDP's.

   Briefly, there are essentially two ways in the literature which one can account for risk sensitivity in MDP's while using a discount factor. The first way uses specific single-period utility functions $U_S$, while the second uses multi-period utility functions $U_M$ that permit exact DP recursion.

   In this chapter, we propose a parameterized structure concerning a single-period risk

sensitive linear utility function that can be used within exact DP and ADP algorithm under the classic Bellman equations. This parameterized structure tries to approximate the exact multistage expected performance-risk tradeoff. We test this utility on stochastic shortest path instances against an multi-period exponential utility.

6. Finally in Chapter 7, I summarize the work and present some possible extensions for future work.

# CHAPTER 2

# BACKGROUND

This chapter provides the background information on which this work is built, and reviews related work from the literature. In Section 2.1, we go over the bottlenecks of mathematical programming when applied to generic multistage problems. Furthermore, Section 2.1 attempts to communicate the value of our approach, showing how mathematical programming advances can be applied. Section 2.2 discusses the characteristics of an MDP formulation. In Section 2.3, we delineate the exact DP methodology and theory with respect to its convergence, along with its computational obstacles. In Section 2.4 we go over basic utility theory and review some results for risk sensitive objectives. In Section 2.5, we review state of the art methodologies that address MDPs under risk-sensitive objectives.

## *2.1 Mathematical Programming Applied To Multistage Problems*

Multi-stage decision-making under uncertainty has been approached both through mathematical programming and dynamic programming methods.

Mathematical programming with random variables whose values will be revealed in the future has been a subject of considerable research [6] [19] [20] [21], but runs into several bottlenecks:

1. Solving for an expected value by sampling the future misses the opportunity to revise actions depending on the state.

2. Solving the full problem where actions can depend on the state requires that the branching of the future scenarios be taken into account. This by itself presents the following problems:

    - The number of branching points and scenarios is exponential in the number of

time periods. Therefore even writing the problem as an explicit mathematical program can be very difficult.

- One needs fairly restrictive assumptions about how the actions and the future interact. For example, it is very difficult to express situations in which the actions change the nature of the underlying transitions (e.g., by revealing information.)

Receding or rolling horizon mathematical programming takes care the lack of feedback, but runs into a compromise between the first two problems. Essentially, one can limit the combinatorial explosion using the rolling horizon idea, but this does not solve the problem of having choices now, that depend on the future, which are not properly evaluated. In this thesis, we will compare the developed ADP approaches against rolling horizon mathematical programming strategies to quantify the benefits of ADP in the face of uncertainty.

### 2.1.1 Deterministic Optimization Applied To Multistage Problems

Before communicating in a clear manner the value of the thesis, I will examine the limitations when applying deterministic optimization to important combinatorial optimization problems via a realistic example.

#### *2.1.1.1 Lagragian Decomposition Combined With Sample Average Approximation in Realistic Supply Chain Example*

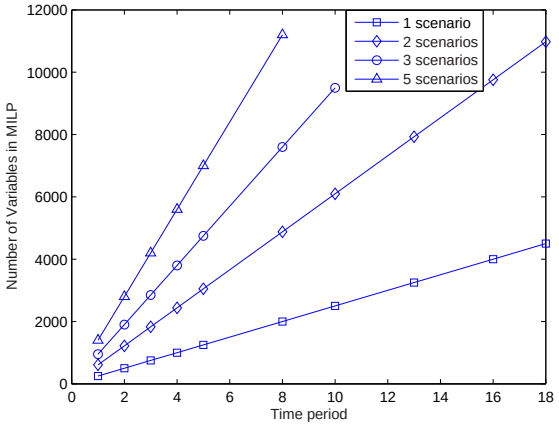
Mixed integer linear program (MILP) formulations are increasingly used in different disciplines (scheduling [22], biological regulatory networks [23] etc.). A relaxation approach to the solution of large integer programming problems is to take a set of "complicating" constraints into the objective function in a Lagrangian fashion (with fixed multipliers that are changed iteratively). This approach is known as Lagrangian relaxation.

Using this technique *Lee et al* [24] and *Pinto and Grossman* [25] address realistic refinery supply chain problems under uncertainty.

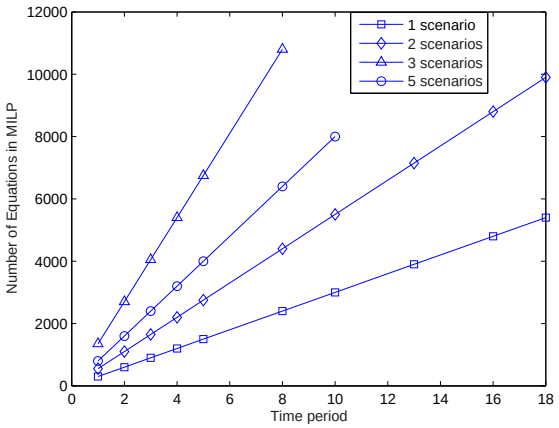
The uncertainty is represented in the form of a set of deterministic scenarios. The model and logic constraints are replicated for each sampled scenario. This produces a large deterministic problem that approximates the exact solution of the multistage stochastic program. Currently important multistage applications are usually addressed via a Sample

Average Approximation (SAA) methodology (e.g. SAA can be applied in routing problems [21], or asset investment problems [26], or even in numerous articles about supply chain design and operation [6]).

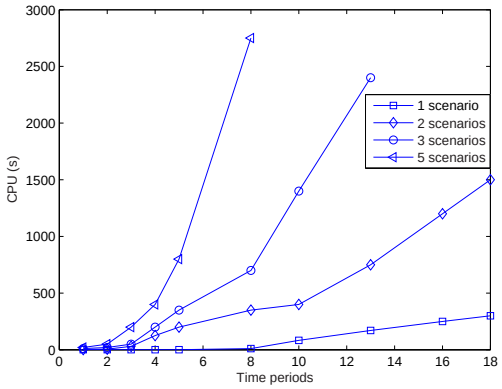
Computational results on a realistic supply chain problem discussed in [24] are summarized at Fig. 3. Given that the number of scenarios increase linearly, so do the variables and the equations of the relaxed MILP (Fig.3 (a),(b) ), but the CPU time, as shown in Fig.3(c), to solve the MILP increases exponentially. This argument-example demonstrates solver limitations when trying to incorporate hundreds or thousands of scenarios. Therefore there is a need for a framework that can use deterministic numerical solutions and construct an actual policy for multistage problems. This provides a motivation when formulating our algorithms.




(a) Number of variables in Pinto's MILP formulation with respect to the number of scenarios that it considers (Pinto and Moro [27]).




(b) Number of equations in Pinto's MILP formulation with respect to the number of scenarios that it considers (Pinto and Moro [27]).




(c) CPU time in Pinto's MILP formulation with respect to the number of scenarios that it considers (Pinto and Moro [27]).

**Figure 3:** Exponential increase of the CPU time for a continuous MILP formulation of a realistic supply chain application with respect to the number of scenarios it considers.

## 2.2 Markov Decision Processes

For the purposes of this thesis, the choice of describing the uncertainty is via a first order Markov chain [11]. Markov chain is a discrete stochastic process that inherits its name from the Markov property [11]. The term Markov property means that the conditional probability distribution of the future system states depends only upon the present state, and not on any past states.

Before proceeding further, some specification and definitions are in order. This thesis mainly considers discounted infinite horizon discrete-time MDP's. The usage of the discount factor $\gamma$ guarantees the convergence of DP, under mild assumptions. Those assumptions are : 1) Finite state space denoted as $\mathbb{S}$, 2) Finite action space denoted as $\mathbb{A}$, 3) Probability space denoted as $\Omega$ with finite support, 4) Bounded single stage rewards/costs $|\hat{f}(s, \alpha, \omega)| < \mathrm{M}$, $\forall (s, \alpha, \omega) \in \mathbb{S} \times \mathbb{A} \times \Omega$.

The dynamics of discrete event systems follow the general system equation, $s_{t+1} = \Phi(s_t, \alpha_t, \omega_{t+1})$. In words, the state of the system at time $t$ + *1* is a function $\Phi$ of the state $s_t$, the decision $\alpha_t$, and a random disturbance $\omega_{t+1}$ that takes place just before time *t+1*.

An MDP is formally defined by a tuple $(\mathbb{S}, \mathbb{A}, \Phi(\cdot,\cdot,\cdot), f(\cdot,\cdot))$, where $f(\cdot,\cdot)$ denotes the one stage expected stage-wise cost/profit function with arguments a state-action pair. Using subscript $t$ to denote the value of any variable at time $t$, and with some abuse of notation, we write $\Phi(\cdot,\cdot,\cdot) : s_t \times \alpha_t \times \omega_{t+1} \rightarrow P(s_{t+1}|s_t, \alpha_t)$, $(\alpha_t \in \mathbb{A})$ where $P(s_{t+1}|s_t, \alpha_t)$ represents the finite support probability distribution of the successive state $s_{t+1}$ given the state-action pair of $s_t, \alpha_t$ under the realization of the uncertainty $\omega_{t+1}$. Furthermore, a decision policy $\pi \in \Pi^{MD} : s \rightarrow \alpha$ is a map indicating what action, $\alpha$, to take for any given system state, $s$. Here $\Pi^{MD}$ represents the set of all admissible Markovian deterministic policies. Because of the mild assumptions and the usage of the discount factor we restrict our attention to such policies and eliminate the chance that an optimal policy can be randomized (described by a probability distribution over the action sets).

## 2.3 Dynamic Programming

I refer the reader to *Putterman* [11] for an excellent introduction on DP. For a very strict mathematical description of DP the reader is referred to *Stokey et al* [28].

### 2.3.1 The Value Function

Very loosely, DP is a methodology that converges to a fixed point the value of an unknown function for all the system states. The converged function is the so called value function. By the term system state we mean the minimum collection of the variables, which completely characterizes the future behavior of the given system.

" What is so particular of this value function?" Simply, by obtaining this function the optimal policy for deterministic or stochastic, one stage or multi stage problems is trivially defined given the necessary assumptions as presented at the previous paragraph. To retrieve this function for discrete event systems, one usually results to numerical methods.

The main feature of DP is the "value function" denoted as $J$ , which maps the system state $s_t$ to its resultant expected *total* discounted reward under some policy $\pi$.

$$J^{\pi}(s_t) = \mathbb{E}\left\{\sum_{t=0}^{\infty} \gamma^t \hat{f}(s_{t+1}, \pi(s_t), \omega_{t+1}) | s_0\right\} \quad , \gamma \in [0,1) \tag{1}$$

Note that I use index $t$ to imply the current state and index $t+1$ for the successive state. $\hat{f}(s_{t+1}, \pi(s_t), \omega_{t+1})$ represents the one stage stage-wise or myopic reward received by exercising $\pi : s_t \rightarrow \alpha_t$ , ($\alpha_t$ is the control) at a given state $s_t$, with the realization of the stochastic variable-uncertainty $\omega_{t+1}$. The discount factor is denoted by $\gamma$. Its purpose is to discount the future rewards to the present.

When using a discount factor, DP in its exact form guarantees the retrieval of a Markovian deterministic optimal policy $\pi = \pi^*$ and the optimal value function, $J^{\pi^*}(s)$ (Eq.(2)).

$$J^{\pi^*}(s_t) = \max_{\pi \in \Pi^{MD}} J^{\pi}(s_t) \tag{2}$$

The optimal value function $J^{\pi^*}(s_t)$ satisfy the Bellman or optimality equation. More information about the properties of the optimality equations can be retrieved in [11, 29].

$$J^{\pi^*}(s_t) = \max_{\alpha \in \mathbb{A}} \mathbb{E}\{\hat{f}(s_{t+1}, \alpha, \omega) + \gamma \sum_{s_{t+1} \in \mathbb{S}} P(s_{t+1}|s_t, \alpha) J^{\pi^*}(s_{t+1}) | S_t = s_t\} \tag{3}$$

By $S_t = s_t$, we imply that the current state is a random variable $S_t$ and $s_t$ is one of its realizations. The optimality equation is always conditioned on the current state realization $s_t$.

If the state space is continuous and has some specific structure, linear dynamics and we use quadratic cost function and Gaussian disturbances, then there are analytical expressions (algebraic Ricatti equations [3]) that provide the optimal control (optimal policy). The Ricatti equations work under some additional restrictions 1) that the system is controllable, that means that the control policy affects the system states and 2) the choice of the quadratic function must be proper, meaning that it should minimize the system's unstable modes [3]. This sort of formulation is very popular in the process control systems engineering community where the state space is continuous.

In general the optimal value function $\forall s \in \mathbb{S}$, solves the stochastic multistage decision making problem. The optimal value function is obtained through value iteration (VI) or policy iteration (PI) or exact linear programming (detail description,convergence proof and analysis as well as small numerical examples of these techniques can be found in *Putterman* [11] and *Bertsekas* [29]). I present the value iteration algorithm in the next session.

### 2.3.2 The Value Iteration Algorithm

The description of the classical value iteration algorithm follows:

For a given problem with a finite state space $\mathbb{S}$, action space $\mathbb{A}$ and probability space $\Omega$:

1. Initialize the value function $J^0(s_t)$ $\forall s \in \mathbb{S}$

2. Iterate over all states $s_t \in \mathbb{S}$ for all actions $\alpha_t \in \mathbb{A}$ using the Bellman equation Eq.(4).

$$J^{i+1}(s_t) = \max_{\alpha \in \mathbb{A}} \{ f(s_{t+1}, \alpha_t) + \gamma \sum_{s_{t+1} \in \mathbb{S}} P(s_{t+1}|s_t, \alpha_t) J^i(s_{t+1}) \} \tag{4}$$

$$f(s_{t+1}, \alpha) = \mathbb{E}_\omega [\hat{f}_{t+1}(s_{t+1}, \alpha_t, \omega_{t+1}|s_t))]$$

i : is the iteration index note that the entire state space is swept through in each iteration.

3. Terminate Bellman iteration when : ($\|J^{i+1}(s_t) - J^i(s_t)\|_\infty < \varepsilon$ ), where $\varepsilon$ is a problem specific constant.

With the converged value function $J^*$, the optimal control for a given state is given according to Eq.(5).

$$\alpha^* = \mathbf{arg}\max_{\alpha \in \mathbb{A}} \{f(s_{t+1}, \alpha) + \gamma \sum_{s_{t+1} \in \mathbb{S}} P(s_{t+1}|s_t, \alpha) J^i(s_{t+1})\} \tag{5}$$

The value iteration is a **monotonic** linear operator and guarantees the convergence to a fixed point (optimal value function) [11]. In mathematics such an operator is called contraction mapping. This means that no matter the initialization (overestimation or underestimation) the value iteration will converge to the same fixed point after a finite number of iterations. The mathematical proofs about the convergence of the DP to a fixed point solution and the properties of the DP operator will follow at Section 2.3.4.

However, value iteration is compromised by the so called 'curse of dimensionality (COD)'. This refers to the proportional growth in the computational load with respect to $|\mathbb{S}|$ and $|\mathbb{A}|$, where the $|\cdot|$ operator represents set cardinality. For instance, the computational load per a single VI iteration of a fully connected graph scales as $|\mathbb{S}|^2|\mathbb{A}|$. As the cardinality of the state and action space tend to grow exponentially with the dimensions of the state and action variables, the computational requirement can quickly become unwieldy. For almost all problems of practical interest, $|\mathbb{S}|$ and $|\mathbb{A}|$ are too large to admit these exact DP approaches. Therefore, the use of discretization and interpolation schemes is unavoidable, with the caveat that convergence can no longer be guaranteed [30].

The other exact dynamic programming algorithms are the policy iteration and the linear programming approach. Both of these are delineated in Putterman [11].

### 2.3.3 The Dynamic Programming Operator

At this section I formally prove important properties of the DP operator denoted as $T$.

$$(TJ)(s_t) = \max_{\alpha \in \mathbb{A}} \sum_{s_{t+1} \in \mathbb{S}} P(s_{t+1}|s_t, \alpha_t)(\hat{f}(s_t, \alpha, \omega_{t+1}) + \gamma J(s_{t+1}))$$

Also let's define the operator $T_\pi$ with respect to a fixed policy $\pi$:

$$(T_\pi J)(s_t) = \sum_{s_{t+1} \in \mathbb{S}} P(s_{t+1}|s_t, \alpha_t)(\pi(s_t))(\hat{f}(s_t, \alpha, \omega_{t+1}) + \gamma J(s_{t+1}))$$

Now let's prove an interesting property of the operator and the optimal value function $J^*$.

**Theorem 1.**

$$J^* = \lim_{N \to \infty} T^N J.$$

*Proof.* Let's look at $J_\pi(s_0)$ and split up the expectation in it in two parts:

$$J_\pi(s_t) = \mathbb{E}\left[\sum_{t=0}^{N-1} \gamma^t \hat{f}(s_t, \pi_t(s_t), \omega_{t+1}) \middle| s_0 = s_t\right] + \mathbb{E}\left[\sum_{t=N}^{\infty} \gamma^t \hat{f}(s_t, \pi_t(s_t), \omega_{t+1}) \middle| s_0 = s_t\right]$$

Let's look at the second term. Notice that its absolute value is less than $\frac{\gamma^N}{1-\gamma} M$, where M is a constant such that $|\hat{f}(s_t, \pi_t(s_t), \omega_{t+1})| < M$.

Recall that

$$(T^N J)(s_0) = \min_{\pi_0, \ldots \pi_{N-1}} \mathbb{E}\left[\sum_{t=0}^{N-1} \gamma^t \hat{f}(s_t, \pi_t(x_t), \omega_{t+1}) + \gamma^N J(s_N) \middle| s_0\right]$$

Now using our bound on the absolute value of the second term, and the above, we can write the following inequalities:

$$J_\pi(s_0) - \frac{\gamma^N}{1-\gamma} M - \gamma^N \|J\|_\infty \leq \mathbb{E}\left[\sum_{t=0}^{N-1} \gamma^t \hat{f}(s_t, \pi_t(s_t), \omega_{t+1}) + \gamma^N J(s_N) \middle| s_0\right]$$

$$\leq J_\pi(s_0) + \frac{\gamma^N}{1-\gamma} M + \gamma^N \|J\|_\infty$$

Let's minimize each term w.r.t $\pi$:

$$J^*_\pi(s_0) - \frac{\gamma^N}{1-\gamma} M - \gamma^N \|J\|_\infty \leq T^N J \leq J^*_\pi(s_0) + \frac{\gamma^N}{1-\gamma} M + \gamma^N \|J\|_\infty$$

Clearly as $N \to \infty$, $\gamma^N \to 0$. Since $N$ was arbitrary it follows that $J^* = \lim_{N \to \infty} T^N J$.

The above proof used our assumption of finite state space to get an upper bound $M$ on $\hat{f}$. It needs additional assumptions to work with infinite state spaces. □

We can also show that the operator $T$ has the following additional properties:

**Theorem 2.** *(Max-norm contraction) $T$ is a maximum norm $\gamma$-contraction. That is, $\|TJ - T\overline{J}\|_\infty \leq \gamma\|J - \overline{J}\|_\infty$ for all $J, \overline{J}$.*

*Proof.* For arbitrary functions $g, h : A \to \mathbb{R}$, where $A$ is some arbitrary set, the following property holds:

$$\left|\min_a g(a) - \min_a h(a)\right| \leq \max_a |g(a) - h(a)|.$$

Using this property we get

$$\begin{aligned}
|(TJ)(s) - (T\overline{J})(s)| &= \Big|\min_\alpha \left(\sum_{s_{t+1}\in\mathbb{S}} P(s_{t+1}|s_t,\alpha_t)(\hat{f}(s_t,\alpha,\omega_{t+1}) + \gamma J(s_{t+1})\right) \\
&- \min_\alpha \left(\sum_{s_{t+1}\in\mathbb{S}} P(s_{t+1}|s_t,\alpha_t)(\hat{f}(s_t,\alpha,\omega_{t+1}) + \gamma \overline{J}(s_{t+1})\right)\Big| \\
&\leq \max_\alpha \gamma \sum_{s_{t+1}\in\mathbb{S}} P(s_{t+1}|s_t,\alpha_t)|J(s_{t+1}) - \overline{J}(s_{t+1})| \\
&\leq \gamma\|J - \overline{J}\|_\infty.
\end{aligned}$$

Since $\|TJ - T\overline{J}\|_\infty = \max_s |(TJ)(s) - (T\overline{J})(s)|$, the previous inequality implies $\|TJ - T\overline{J}\|_\infty \leq \alpha\|J - \overline{J}\|_\infty$. □

**Theorem 3.** *(Monotonicity) If $J \geq \overline{J}$, then $TJ \geq T\overline{J}$.*

*Proof.* Suppose $J \geq \overline{J}$. Then

$$\sum_{s_{t+1}\in\mathbb{S}} P(s_{t+1}|s_t,\alpha_t)J(s_{t+1}) \geq \sum_{s_{t+1}\in\mathbb{S}} P(s_{t+1}|s_t,\alpha_t)\overline{J}(s_{t+1}) \quad \forall s \in \mathbb{S}, \alpha \in \mathbb{A}$$

By multiplying both sides by $\gamma$ and adding the term $\sum_{s_{t+1}\in\mathbb{S}} P(s_{t+1}|s_t,\alpha_t)(\hat{f}(s_t,\alpha,\omega_{t+1})$ to both sides of the inequality, we get $T_\pi J \geq T_\pi \overline{J}$ for any decision rule $\pi$. Suppose $\pi^*$ is such that $T_{\pi^*}J = TJ$. Then $TJ \geq T_{\pi^*}\overline{J}$. Also, it is clear that $T_{\pi^*}\overline{J} \geq T\overline{J}$. Therefore $TJ \geq T\overline{J}$. □

**Theorem 4.** *(Offset property) Let $e$ be such that $e(s_t) = 1$ for all $s_t \in \mathbb{S}$. Then $T(J + ce) = TJ + \gamma ce$ for all $c \in \mathbb{R}$.*

*Proof.*

$$\begin{aligned} T(J+ce)(s_t) &= max_{\alpha\in\mathbb{A}} \sum_{s_{t+1}\in\mathbb{S}} P(s_{t+1}|s_t,\alpha_t)(\hat{f}(s_t,\alpha,\omega_{t+1}) + \gamma(J(s_{t+1}) + ce(s_{t+1}))) \\ &= max_{\alpha\in\mathbb{A}} \sum_{s_{t+1}\in\mathbb{S}} P(s_{t+1}|s_t,\alpha_t)(\hat{f}(s_t,\alpha,\omega_{t+1}) + \gamma J(s_{t+1})) + \gamma c \\ &= (TJ)(s_t) + \gamma ce(s_t) \end{aligned}$$

□

### 2.3.4 Contractions

As was shown in the previous section, the dynamic programming operator $T$ is an $\gamma$-contraction in the max-norm. In this section we will prove some useful properties of contractions, and discuss some of their implications for dynamic programming. Throughout this section we will let $F$ be a $\gamma$-contraction with respect to some norm $\|\cdot\|$. For simplicity we will assume $F : \mathbb{R}^n \to \mathbb{R}^n$.

**Theorem 5.** *The sequence $\{F^N J\}$ converges for any $J$.*

*Proof.* Since $F : \mathbb{R}^n \to \mathbb{R}^n$, it will suffice to show that $\{F^N J\}$ is a Cauchy sequence. Since $F$ is an $\alpha$-contraction, $\|FJ - F^2J\| \leq \alpha\|J - FJ\|$. In general, $\|F^N J - F^{N+1}J\| \leq \alpha^N\|J - FJ\|$. To show that $\{F^N J\}$ is a Cauchy sequence, we need to show that for any $\epsilon > 0$, there exists some $K$ such that $\|F^M J - F^N J\| \leq \epsilon$ for all $M, N \geq K$. For any $K$ and $M, N \geq K$,

$$\begin{aligned} \|F^M J - F^N J\| &= \left\| \sum_{i=M}^{N-1} (F^i J - F^{i+1} J) \right\| \\ &\leq \sum_{i=M}^{N-1} \|(F^i J - F^{i+1} J)\| \\ &\leq \sum_{i=M}^{N-1} \alpha^i \|(J - FJ)\| \\ &\leq \frac{\alpha^K}{1-\alpha} \|(J - FJ)\| \end{aligned}$$

For any $\epsilon > 0$, we can find $K$ such that

$$\frac{\alpha^K}{1-\alpha} \|(J - FJ)\| \leq \epsilon,$$

hence $\{F^N J\}$ is a Cauchy sequence. □

**Theorem 6.** *$F$ has a unique fixed point.*

*Proof.* The sequence $\{F^N J\}$ converges to a fixed point of $F$, so at least one fixed point exists. Now suppose $J_1$ and $J_2$ are both fixed points of $F$. Since $FJ_1 = J_1$ and $FJ_2 = J_2$, this implies

$$\|FJ_1 - FJ_2\| = \|J_1 - J_2\|,$$

contradicting the contractive property of $F$. Therefore, the fixed point of $F$ is unique. □

Recall that the dynamic programming operator $T$ is a max-norm $\alpha$-contraction and that $T^N J \to J^*$ as $N \to \infty$. By the previous two theorems, we can conclude that $J^*$ is the unique solution to the equation

$$J^* = TJ^*.$$

This is known as *Bellman's equation*. We can also use the fact that $T_\mu$ is a max-norm $\alpha$-contraction for any $\mu$ to establish the following result:

**Theorem 7.** *A* stationary policy $\pi = \{\mu, \mu, \mu, \ldots\}$ *is optimal among all policies if and only if* $TJ^* = T_\mu J^*$.

*Proof.* First suppose that the stationary policy described by $\mu$ is optimal. Let $J_\mu$ be the cost-to-go function under this policy. Since this policy is optimal, $J^* = J_\mu$. Also, the equation $J = T_\mu J$ is uniquely solved by $J_\mu$. So $J_\mu = T_\mu J_\mu \implies J^* = T_\mu J^* \implies TJ^* = T_\mu J^*$. Now suppose $TJ^* = T_\mu J^*$. This implies $J^* = T_\mu J^*$. Since $J_\mu$ is the unique solution of the equation $J = T_\mu J$, $J^* = J_\mu$, so the stationary policy described by $\mu$ is optimal. □

### 2.3.5 A Review Of Approximate Dynamic Programming Techniques

The evolving stream of the ADP literature aims to develop conceptual frameworks that reduce the computational obstacles of full DP and achieve a high quality solution. The first and most significant source of computational burden is associated with $|\mathbb{S}|$. The two other sources of computational bottleneck are $|\mathbb{A}|$, and the calculation of the expectation operator within the maximization (or minimization) as seen in Eq.(4).

#### *2.3.5.1 Minimizing the COD concerning* $|\mathbb{S}|$

Most of the approaches proposed by the Artificial Intelligence community attempt to minimize $|\mathbb{S}|$ by intelligently sampling the state space and then building a function approximator based on the recorded values of the sampled states. This raises both the question of how to sample the space and what approximator to use. The naive approach of employing uniformly spaced sampling of the state space is not attractive because it is subject to an exponential growth with respect to the state dimension. A more efficient discretization scheme derived from the field of statistical design is orthogonal arrays (OA) based on Latin hypercubes [31]. This sampling scheme has been successfully used as a part of stochastic DP method to solve a high dimensional waste-water treatment planning problem[32].

Another way to sample the state space is to simulate the system under some a priori available policies, such as heuristics and deterministic optimal policies, and sample a finite set of states from the simulated trajectories for constructing the value table. *Lee et al* [33] investigated the performance of such an approach using two different types of function approximators: feedforward neural networks and a non-parametric local approximator called $k$ nearest neighbor averager. Their conclusion was that the use of the local approximator resulted in consistent and stable behavior over the value iteration, whereas the neural networks could display unstable behavior. This finding was supported by empirical results from case studies involving a Van de Vusse Reactor and a MMA polymerization reactor. The test systems are characterized by highly non linear dynamics, which are common in chemical engineering applications.

In similar spirit, *Choi et al.*[18] used the classical value iteration algorithm within a small subset of state space, built by sampling simulated trajectories under various heuristic policies. Their objective was to build a policy or a sequence of policies that improve upon the starting policies. The set of sampled states for which the value iteration was performed represented only a tiny fraction of the entire state space, and hence the major source of the COD was removed. The approach's major limitation is that it does not address the COD associated with the action space. Hence, it only works for MDPs with a relatively small action space, which was the case for their problems.

Another branch of ADP that has attracted significant attention lately is the approximate linear programming based approach suggested by *De Farias and Van Roy*[34]. To solve a DP exactly via linear programming, one must add a constraint for each state-action pair. Alternatively, one can solve its dual problem, where the number of constraints are the same as the cardinality of the sampled state space but the number of decision variables are the feasible actions for each state . *De Farias and Van Roy* [34] suggest the use of sampling via simulation to identify a closed set of states, and subsequently a finite number of basis functions to parameterize the value function, in order to reduce the number of variables. Solving the LP optimizes the coefficients of the chosen basis functions. However, it is not clear how the basis functions should be chosen in the first place in order to achieve solutions close to the optimal one. The selection procedure is heuristic, and it is difficult to decide on the right complexity of the value function surface without over-fitting the function. This approach has been applied to queuing problems where the number of state-action pairs was $10^{12}$ times the cardinality of the action space (about 18).

ADP methods that can address extremely large applications (problems with up to $10^6$ dimensional state space) have been developed for a class of dynamic resource allocation problems [35] [36]. One innovation the authors introduced is to rewrite the Bellman equation using a post-decision state variable, since doing so will lessen the COD associated with the evaluation of the expectation operator. The update still requires evaluation of the expectation over the uncertain variables. However, the authors implement forward dynamic programming, and the approach requires one specific realization of the exogenous uncertainty via sampling. This way the expectation operator can be dropped. The value function approximation strategies include separable, piecewise-linear functions, linear functions, or other basis functions, for which the coefficients are to be estimated. To update the coefficients of theses schemes, the authors utilize a stepsize (learning rate) factor in order to avoid potential outliers and smooth the estimation. More details about the overall approach can be found in [36]. We note that, for these types of problems, their method gave impressive empirical performance results compared to a MIP solution with full information.

#### *2.3.5.2 Minimizing the COD concerning* $|\mathbb{A}|$

The second source of the COD concerning $\mathbb{A}$ was first tackled by *MacQueen* [37], where the optimal value function was bounded to eliminate some suboptimal actions. This action elimination technique has been used in VI and in PI (details in [11, 29]). There is a recently published approach that addresses effectively this issue. This approach is named *evolutionary random policy search* (ERPS) [38]. It turns out that the structure of the Adaptive Action Set (AAS), that we analyze at a later section is very similar to the set of actions proposed in ERPS. The main idea of the Adaptive Action Set (AAS), is that for each state, the value function update will be restricted to only a small set of actions.

Their methodology works on the basis of evolutionary policy iteration. In ERPS they prove that at each iteration the policy does not deteriorate and the learned policy converges to the optimal policy with probability one. There are 3 main differences between the RTADP algorithm that we will propose and the ERPS: 1) The proposed approach constructs a value table with an increasing number of entries starting from an empty value table, while the ERPS needs to initialize a fixed number of states; 2) in the RTADP the exploration rate can be tuned via initialization of the value function, whilst in the ERPS the exploration rate is fixed from the beginning; 3) the RTADP is based on asynchronous value iteration, while ERPS on the policy iteration.

#### *2.3.5.3 Minimizing the COD concerning the expectation operator*

The classical way of circumventing this computational obstacle is to use Monte Carlo sampling, while evaluating each decision. We will implement this idea in our approach later, therefore it will be extensively discussed.

A different very attractive perspective that addresses the same issue is to interchange the expectation with the maximization operator at the traditional Bellman equation by posting these equations around a post state variable. Post state variable is the state of the system after a decision. A detailed description about this notion can be retrieved in [1].

#### *2.3.5.4 Value Function Approximators*

As an endnote for this paragraph, we must admit that from a theoretical point of view, accurate approximation of the value function in high dimensional applications is an impossible task [39]. Approximating the value function surface with simplistic basis functions, e.g. piecewise linear functions, is definitely a rough compromise, and is justified only if one knows a-priori that the optimal value function structure [1]. Even if one uses the non parametric $k - NN$ approach is subject to the fact that all sample points are close to an edge of the sample. To be more specific consider 500 data points uniformly distributed in a 10- dimensional ball centered at the origin. The mean distance to the nearest point is $\approx$0.52, this means the data are closer to the boundary than to the point, hence one must extrapolate from a neighboring sample and not interpolate! That example shows that our logic collapses, when dealing with high dimensional spaces. Extensive discussion of local methods in high dimensions can be retrieved in *Hastie et al* [39].

The issue of structure of the optimal value function is critical. In fact to derive such a structure may be the only way to devise fast algorithms for large MDPs. To speculate and use complex approximation schemes for high dimensional spaces that update the coefficients of localized basis function, like multiple adaptive regression splines [32] is currently the best bet to achieve a good online performance. Nonetheless, this sort of an approximation cannot produce any mathematical guarantees for the convergence of the value function surface.

## *2.4 Risk Measures*

To analyze and account for risk due to uncertainty in decision-making, the adoption of a quantitative measure for risk is required. Such a measure should not lead to counter-intuitive outcomes. For example a risk measure ought to capture that portfolio diversification should lead to risk reduction, not an increase. To ensure such intuitive rules *Artzer et al*[40] defined in their seminar paper the class of 'coherent' risk measures as those that satisfy four main axioms, which are sub-additivity, monotonicity, positive homogeneity, and translation

[1] In Partially Observable Markov Decision Processes (POMDP) one knows that the optimal value function is piecewise linear and convex.

invariance.

Let's proceed to the formal definitions: Consider a random outcome $Z$ viewed as an element of a linear space $\mathbb{Z}$ of measurable functions, defined on an appropriate sample space. According to the seminar paper of *Artzer et al* [40], a function $\rho : Z \to \mathbb{R}$ is said to be a coherent risk measure for $\mathbb{Z}$ if it satisfies the following axioms:

1. Convexity: $\rho(a_1 Z_1 + (1 - a_1)Z_2) \leq a_1\rho(Z_1) + (1 - \alpha_1)\rho(Z_2) \forall Z_1, Z_2 \in \mathbb{Z} and a_1 \in [0, 1]$
2. Monotonicity: If $Z_1, Z_2 \in \mathbb{Z}$ and $Z_2 > Z_1$ then $\rho(Z_2) > \rho(Z_1)$
3. Translation Equivariance: If $\alpha_1 \in \mathbb{R}$ and $Z_1 \in \mathbb{Z}$ then $\rho(Z_1 + \alpha_1) = \rho(Z_1) + \alpha_1$
4. Positive Homogeneity: If $\alpha_1 > 0$ and $Z_1 \in \mathbb{Z}$ then $\rho(\alpha_1 Z_1) = \alpha_1\rho(Z_1)$

Commonly used risk measures like standard deviation ($\sigma$) or Value at Risk ($V\mathrm{a}R$) violate at least one of these properties and therefore can lead to counter-intuitive outcomes in certain situations. In simple terms those measures are not appropriate measures for risk. Numerical data sets that force these measures to violate specific axioms, as well as a more intuitive in interpretation of the axioms , can be found in [41].

$V\mathrm{a}R_\alpha$ is usually meant for loss distributions and corresponds to an upper percentile dictated by the confidence interval $\alpha$. For instance, $V\mathrm{a}R_{95\%}$ is an upper estimate of losses which is exceeded with 5% probability.

The formal definitions of the $V\mathrm{a}R_\alpha$ a for loss distributions follow:

$$VaR_\alpha = max\{\zeta \in \mathbb{R} : (P(z) \geq \zeta) \leq \alpha\}$$

As an alternative to the popular ($VaR_\alpha$), a coherent risk measure called Conditional Value at Risk ($CV\mathrm{a}R$) has been proposed in the recent risk literature. $CV\mathrm{a}R_\alpha$ is defined for an arbitrary profit distribution $f$ as: $CV\mathrm{a}R_\alpha = \mathbb{E}[f|f < VaR_\alpha]$, which represents the mean of the tail of the $(\alpha) \times 100$ bottom percentile of the distribution. In the above $V\mathrm{a}R_\alpha$ represents the cut-off value for the corresponding percentile. The most attractive characteristics of the $CVaR_\alpha$ measure are: a) consistency with the mean-variance [42] approach in one stage problems for normal loss distributions, b) convexity leading to an

attractive one stage optimization problem via LP even for non-normal distributions, and c) the capacity to handle fat tails (e.g., Student-T distributions).

## *2.5 Pareto Optimal Frontier - Efficient Frontier*

Pareto optimality, named after Italian economist Vilfredo Pareto, is a measure of efficiency in multi-objective and multi-party situations. The concept has wide applicability in economics, game theory, multi-objective decision-making, and the social sciences generally. Multi-objective problems consider naturally two or more objective. Sometimes these objectives are measured in different units and no agreed-upon conversion factor exists to convert all criteria into a single metric.

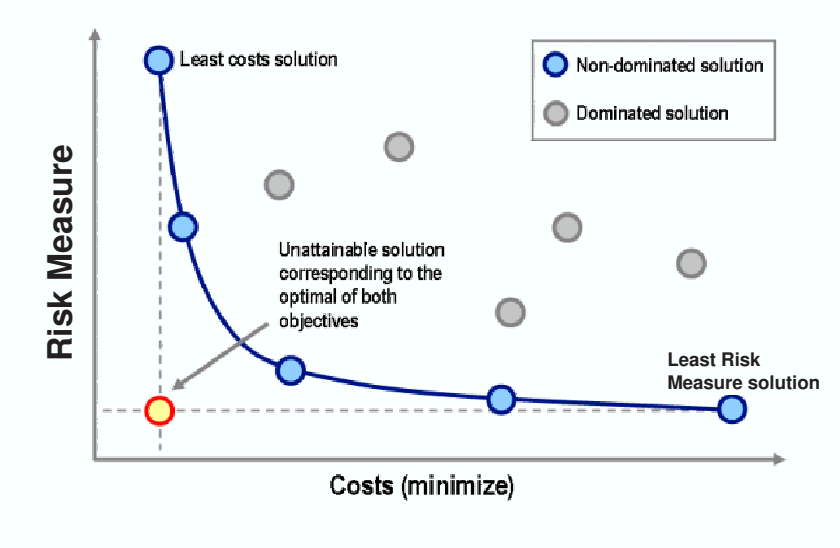



**Figure 4:** Pareto Optimal Frontier - Efficient Frontier.

Pareto optimality can be visualized in a scatter-plot of solutions (see Fig.5). Each criteria (or objective function) is graphed on a separate axis. It is easy to visualize in a problem with only two objectives, but much more difficult with three or more objective. For the purposes of this thesis, we will focus on two objectives. To minimize/maximize the system's expected cost/profit and its $CVaR_\alpha$. It will be seen at the sixth chapter that minimizing/maximizing the $CVaR_\alpha$ corresponds to manipulating the standard deviation of the semi-variance of the worst rather than the expected mean cases.

In a problem with two objectives, both of which are to be minimized Pareto-optimal solutions are those in the scatterplot with no points down and to the left of them. This scatterplot is the so called efficient frontier. Given a multistage problem each of these points would correspond to the expected performance and risk measure of a generated policy.

## *2.6 Expected Utility Decision Theory*

The expected utility hypothesis of *Neumann and Morgenstern* [43] is the cornerstone of utility theory. It axiomatizes this hypothesis in terms of agents' preferences via binary preference relations over different actions given the uncertainty, and therefore is a natural fit for optimization.

Currently, there is difficulty applying utility theory within the context of discounted multi-stage problems. A key task of this thesis is to device a mechanism that would generate such solutions.

Assume $X$ and $Y$ are the stochastic reward processes $X = \{x_1, x_2, ...\} \in V$ (where $V$ is the set of all possible reward processes) and $Y = \{y_1, y_2, ...\} \in V$ created by two distinct policies. In this context, we want to retrieve a policy which will create a stochastic process of state transitions and corresponding rewards and be able to express risk-time preference. The objectives or criteria that will allow us to distinguish with a risk-time preference ordering on $V$ that $X \succeq Y$ ($X$ is at least as acceptable as $Y$) are the following:

In summary, the criteria that we will use to express risk preferences in multi-stage problems will be:

- Criteria 1: The following statement represents risk neutrality and risk sensitivity, respectively, concerning multi-period utility and single-period utility and it is well justified only for deterministic system because of Koopman's [44] [45] [46] axiomatic foundation.

$$X \succeq Y \Leftrightarrow \mathbb{E}\big[\sum_t \gamma^t U_S(X_t)\big] \succeq \mathbb{E}\big[\sum_t \gamma^t U_{Intra}(Y_t)\big] \qquad ?$$

  This preference does not have an axiomatic foundation that will generate time-risk

sensitive policies when applied to multistage stochastic systems. Nonetheless, in chapter 6 we will demonstrate that given a specific structure for this $U_S$, we will be in position to well approximate at least a portion of the true multi-stage efficient frontier.

- Criteria 2: The following statement expresses risk sensitivity and risk neutrality, respectively, concerning multi-period utility and single-period utility and at present it presents the only logical formalism that expresses time-risk preferences.

$$X \succeq Y \Leftrightarrow \mathbb{E}\big[U_M(\sum_t \gamma^t X_t)\big] \succeq \mathbb{E}\big[U_M(\sum_t \gamma^t Y_t)\big]$$

This two criteria coincide for one stage problems. More details about accounting time-preferences within DP are delineated at the sixth chapter.

The formalism that would allow to retrieve a risk sensitive multistage policy should regard the summation of the corresponding Certainty Equivalence (CE) as the value function *Chung and Sobel* [12]. The only class of functions that allows to write recursive equations based on that fact are exponential functions, since their inverse are logarithmic functions. The optimality equations based on this exponential utility are well derived by *Chung and Sobel* [12] and by *Avila-Godoy* [47].

Let's delineate the concept is the CE. The possible types of decision makers are : 1) risk-seeking, 2) risk-averse, and 3) risk-neutral. The shapes of the utilities that represent those types of decision makers are shown at Fig.4. A CE represents the maximum amount of money we are willing to pay for some gamble. Alternately, a CE is the minimum premium we are willing to pay to insure us against some risk. Imagine that a gambler offers you the following bet: If a fair coin lands heads you will lose \$1000, but if it lands tails he will award you \$1000. How much each of the three decision makers (Fig.4) would you be willing to pay for this chance?

The expected utility from this gamble is half-way between the utility from winning \$1,000 and losing \$1,000, since each event is equally likely. The risk taker will be willing to pay something less from 500 dollars for this gamble. The risk neutral is willing to pay \$0 for this gamble, while the risk averse decision making would want to receive something

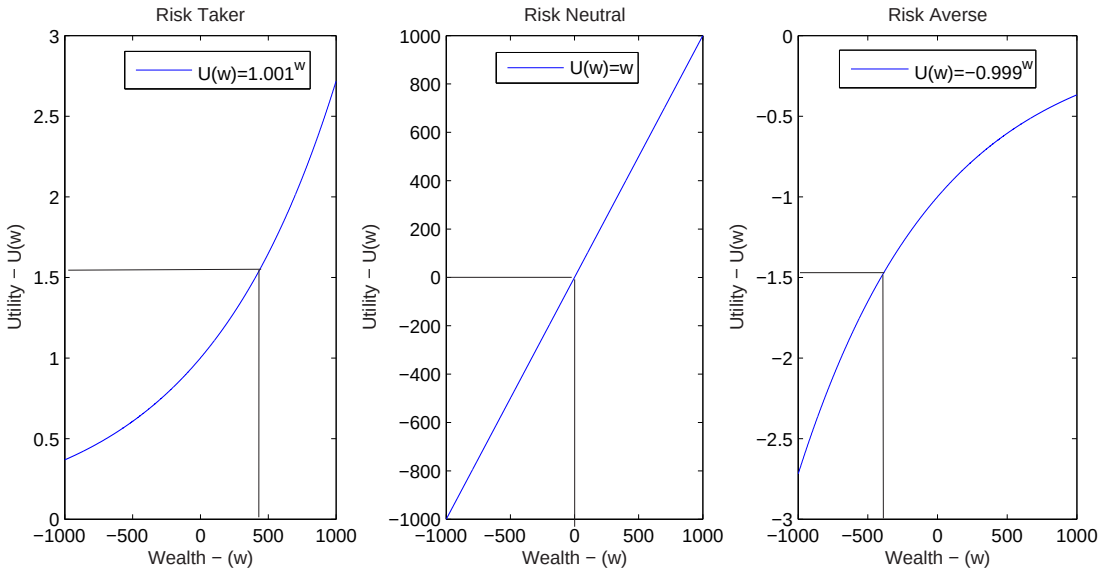



**Figure 5:** At the left this convex utility expresses a risk taking attitude. At the middle this linear utility expresses a risk neutral attitude. At the right this concave utility expresses a risk averse attitude.

less than $ 500 for this gamble as insurance. The more risk-averse a person is, the lower is his/hers certainty equivalent.

A different way to express risk sensitive attitudes is via single-period linear utilities. This traces back to 1989 and 1994, where *J. Filar et al* [48] is concerned in finding policies in discounted and undiscounted variance penalized MDP's and *Sobel M.* [13] in undiscounted MPD's. It turns out that the discounted MDP is more difficult to analyze than the average reward model in the context of the variance-penalized problem. One of our contributions lies in chapter 6, since we will research the question: how can one approximate the true multi-period mean-CVaR utility function using a linear combination of single stage single-period mean-CVaR utilities.

## *2.7 Methodologies Addressing Multistage Risk*

In this section, we will review methodologies that can generate Pareto optimal solution in multi-objective stochastic optimization problems. The reviewed methodologies are: a) the mathematical programming approach, b) the simulation based optimization approach, c)

the Dynamic Programming approach.

### 2.7.1 Mathematical Programming And Simulation Based Optimization Methodologies On Pareto Efficiency

The usage of mathematical programming is very popular when hedging the risk in financial applications. First, *H. Markowitz* [42, 49] proposed portfolio selection via mean-variance analysis in capital markets. He proposed a quadratic programming formulation for single period optimizing the covariance and the profit. *Rockafellar and Uryasef* [50] proposed an LP formulation that optimizes simultaneously the $CVaR_\alpha$ and the $VaR_\alpha$. This LP formulation is equivalent to the mean variance approach when dealing with normal distributions, but it is a superior optimization formulation when dealing with cost/profit distributions that exhibit heavy tails. An extension of this LP optimization to a multi stage portfolio optimization problem is proposed via external sampling by *Borgan* [51]. The interesting feature of their work is that they utilized various pdf's for each instrument, in order to demonstrate the superiority of the $CVaR$ as a risk measure against the traditional variance.

A significant series of papers that investigate how a multistage risk measure can be used within: a) a simulation - optimization DP framework , b) multistage programming are the following by *Cheng et al* [52] [53] [54]. One of their key contributions is that they propose the expected downside risk as a risk measure. This measure needs to satisfy the properties of separability and monotonicity [55], in order to be able to be decomposed into stage-wise separable functions and be used in the DP recursion. In order to achieve that they introduce a auxiliary state variable. This state space augmentation comes in complete agreement with the one proposed by Liu and Koevig [56]. Liu and Koevig [56] proposed a state space augmentation, which will allow the usage of the optimality equations regardless the chosen utility function.

A different contribution derived from this series of papers *Cheng et al* [52] [53] [54] is that they compare numerically pareto efficiencies between a simulation based optimization approach and a multi-stage stochastic mathematical programming approach. The math program provided higher quality pareto solutions, but suffered from practicality issues in comparison with sim-opt approach.

Recent work on risk for multi-stage stochastic problems on capacity expansion and NPD can be retrieved by the research group of Professor Reklaitis [57] [58] [59] [60]. They nicely explain, that a way to account for the multistage risk that characterizes a given policy is to simulate it until the end of the horizon multiple times. To circumvent the COD they introduce a state-action pseudo-utility function, in order to propagate back the optimal decision via simulation. They utilize the separable downside risk measure, and discuss that the majority of the other risk measures are not separable because they lack the separability property [55]. [2]

### 2.7.2 Dynamic Programming Methodologies

The rigorous alternative methodology to mathematical programming, in order to optimize a multistage risk measure is the dynamic programming approach. In Section 2.6, we reviewed the multi-stage optimization criteria and DP methodologies that adopt them.

The next section of this chapter may seem out of context. The reader may express interest in reading the formulation of the following stochastic shortest path problem with explicit start and goal states as an MDP, once he/she comes across this problem at the next chapter. We choose to delineate the formal MDP formulation here, in order to facilitate the thesis flow and familiarize the reader even further with respect to MDP's.

## *2.8 The Formulation Of A Stochastic Shortest Path Problem As An MDP*

As mentioned in the introduction, we initially studied the one dimensional shortest path problem with a single starting and goal state. A high level description of the problem was already given in the introduction. What follows is the formulation of the one dimensional SSP as a formal MDP.

The MDP formulation requires specification of the following elements: State variables, exogenous information variables, decision variables, transition function, one stage cost function, and objective function. The following subsections detail each of these.

[2] The stage by stage propagation cannot provide the necessary information for the risk measure to be backed up

### 2.8.1 State Variables / Exogenous Information Variables

A compact definition of state given in [1] is: *a state variable is the minimal function of history that is necessary and sufficient to model all the future dynamics of the system.* For the SSP problem, the state variable is a one dimensional vector defined as below:

$$\left[ \; s_t = \text{ Grid position at time } t \; \right] \tag{6}$$

The uncertainty is with respect to the state transition and is represented by a sequence of random variables $\omega_1, \omega_2, \omega_3, ...$ with the Markov property, meaning the future state values depend solely on the present state and are independent of the state history. For complex/practical problems like traffic congestion, higher-order Markov models may be used to describe the traffic pattern in a more precise manner. The order selection of the Markov model is highly case-specific and for our study we adopt a first order Markov model. Such models describe in a probabilistic manner the state transitions and under a specific action $\alpha$, are denoted by $P_\alpha$. For example, the $(i,j)^{\text{th}}$ element of $P_\alpha$ is the probability of the transition, taking action $\alpha$, from state $s_i$ to $s_j$ at the next time period. For the classic SSP problem $|\mathbb{A}| = 4$, therefore one needs to store 4 $(n \times n)$ probability transition matrices[3]. A methodology based on the EM algorithm which can systematically identify such matrices from data can be retrieved in [61].

### 2.8.2 Decision Variables

Decisions are modeled in discrete time. The decision space $\mathbb{A}$ encodes all the possible controls that are applicable to each system state $s_t$. Each action or control is concerned with moving the system position at the discrete grid.

$$\left[ \; \alpha_t = \text{ North , South , East , West } \; \right] \tag{7}$$

### 2.8.3 Transition Function

The transition function as explained before is probabilistic:

$$s_{t+1} = P_{\alpha_t} = P(s_{t+1}|s_t, \alpha_t) \tag{8}$$

[3] $|\mathbb{S}| = n$

Since the stochastic transition is independent, the transition to the next state follows the conditional probability distribution $P_{\alpha_t}$ and hence is dictated by the stochastic outcome of a corresponding biased coin. The possible outcomes for the state transition are 4.

### 2.8.4 Contribution (Cost) Function

The one step cost produced by a decision $\alpha_t$ at state $s_t$ during one time period with random variable $\omega_{t+1}$ is denoted as $\hat{f}(s_t, \alpha_t, \omega_{t+1})$. Then, the expression for $\hat{f}(s_t, a_t, \omega_{t+1})$ is:

$$\hat{f}(s_t, a_t, \omega_{t+1}) = C(s_{t+1}) \tag{9}$$

Where $C(s_{t+1})$ is the cost incurred based on the successive state.

The expectation of the one step profit is defined over the probability space $\Omega$:

$$f(s_t, a_t) = \mathbb{E}[\hat{f}(s_t, a_t, \omega_{t+1})] = \sum_{j=1}^{N} P(s_j|s_t, \alpha_t)\hat{f}(s_j, a_t, \omega_j) \tag{10}$$

where $P(s_j|s_t, \alpha_t)$ is the probability of the random variable $\omega_{t+1}$ to be taking the specific realization of $\omega_j$ and $N$ is the number of transitions with non-zero probability conditioned on $s_t$.

### 2.8.5 Objective Function

Usually the primary objective in this problem is to find the policy $\pi$ that minimizes the discounted expected cost over an infinite horizon.

$$\pi^* = \mathbf{arg}\min_{\pi} \left\{ J^{\pi}(s_0) = \mathbb{E}\{ \sum_{t=0}^{\infty} \gamma^t f(s_t, \pi(s_t)) \mid s_0 \} \right\} \tag{11}$$

This goal is accomplished when we consider the entire state space and construct a stationary decision function $\pi : s_t \rightarrow a_t$ such that each state is mapped to the best possible action.

# CHAPTER 3

# A REAL TIME APPROXIMATE DYNAMIC PROGRAMMING APPROACH

The Value Iteration (VI) as presented at the previous chapter is termed "synchronous", since we iteratively need to update or backup the value function of every state of the state space to guarantee convergence to the optimal value function. VI is a rather"safe" algorithm to converge the value functions, meaning that one can still converge by performing less computation. Variants of the VI that achieve this are termed asynchronous VI methodologies, where the user uses some prioritizing rules to select the next state to be updated and result to a faster convergence. Specifically, the convergence of model based reinforcement learning for MDPs was shown by *Gullapalli and Barto* [62]. . A type of asynchronous value iteration scheme is the episodic learning Real Time Dynamic Programming approach as proposed by *Barto et al* [63]. The RTDP approach utilizes the dynamic programming operator to pick an $\epsilon$ greedy action, by fully evaluating the expectation operator using information from the known probability distribution and then selects randomly a successive state for the system to visit. A significant requirement for the RTDP to guarantee convergence is an initial overestimation of the value function for all the states (if its a maximization problem). Our approach is a variant of the RTDP algorithm inspired by the following ideas.

In the following papers [17, 18, 64], researchers show only a tiny fraction of the state space needs to be involved in constructing high quality policies. Ideally, this set of states would correspond to a tight superset of all the states visited under the true optimal policy for the given system. The reason for this can be explained as follows. Let us define the set of states that can potentially be visited under the true optimal policy as the "relevant portion of the state space," which is to be denoted as $\mathbb{S}_R \subseteq \mathbb{S}$. With $\mathbb{S}_R$ being a closed set (i.e., there is an action, the one assigned by the optimal policy, that keeps the state within $\mathbb{S}_R$), it is sufficient to perform value iteration only with $\mathbb{S}_R$ to retrieve the optimal policy. Such

value iteration is termed approximate, since it is not performed on the entire state space. During the approximate value iteration, the user: 1) assigns pessimistic value functions to the states that are far distance wise from the ones sampled, 2) utilizes parametric or non-parametric value function approximations for the states that are close to the ones sampled and are needed for the necessary value function backups. The measure of distance between the states depends on the application and can be a Frobernius norm or the Euclidean norm or the one norm, etc. I utilize such ideas to the classic RTDP to discourage the state space exploration.

In the proposed approach the value table starts with one entry (initial state). The mechanism of building the simulated value table, denoted as $\mathbb{S}_{Sim}$, is typically through simulation. The proposed approach eventually focuses the computation on the states belonging to $\mathbb{S}_{Sim}$, similarly as in asynchronous dynamic programming [15]. The key issue is to estimate the values of the unvisited so far states. The proposed method follows a similar logic as the above approximate value iteration techniques. For those unvisited states with no "nearby" neighbors in the value table, we assign a pessimistic value that discourages further exploration. For those states that have a sufficient number of registered neighbors, their values are estimated using the non-parametric $k$ nearest neighbor $(k-NN)$ averager. Traditionally exploration, which is clearly necessary for real-time methods, is performed by not enforcing 100% of the time the $\epsilon$ greedy control. Here, our intention is to minimize the exploration and to force the computations only to specific portions of the state space.

In the RTADP algorithm , the exploitation is done with respect to the following controls: 1)random actions, 2) heuristic policies, 4) mathematical numerical actions or 5) best known stored action from previous experiences. This way, the state trajectories would be kept within or close to the current set of visited states $\mathbb{S}_{Sim}$. The goal here is to identify a small set of states that would allow the Approximate Value Iteration (AVI) [17] [18] to be computationally tractable, even for systems of large $|\mathbb{S}|$. We note that that this proposed approach has fundamental differences in contrast with the RTDP as first discussed in *Barto et al* [15], which uses an estimator that encourages exploration by assigning optimistic values to the unvisited states and considers the entire state space. However, that RTDP would

lead to the exploration of almost the entire state space, which is clearly impractical for systems with large $|\mathbb{S}|$. It is speculated from the numerical experiments that the proposed approach provides the means for gradually building a simulated set of states $\mathbb{S}_{Sim}$ which approximately encompasses $\mathbb{S}_R$, while iteratively computing value function estimates for the states belonging to $\mathbb{S}_{Sim}$.

Our overall approach is termed Real Time Approximate Dynamic Programming (RTADP), as it is a modified version of Barto's RTDP scheme. The proposed modifications focus on alleviating the COD as a result of large $|\mathbb{S}|$ and $|\mathbb{A}|$. Other potential sources for the COD is the computation needed for evaluating the expectation of the reward and that needed to calculate the transition function, both of which are not addressed in this manuscript as they are not significant for the system of our focus. Overcoming the COD is strictly problem-specific. Unfortunately, there is no general way to eliminate the COD for all problems. Nonetheless the reader is referred to *Powell* [1], since it covers a number of general ways of overcoming the curse of dimensionality by using continuous representations of states and actions, and continuous value function approximations.

The second source of the COD is focused on the cardinality of the action set that needs to evaluated per system state during the value iteration. This issue was addressed by *MacQueen* [37], who proposed some action elimination techniques based on bounds. This action elimination technique has been used in both Value Iteration (VI) and Policy Iteration (PI) (details in [11, 29]). The decision space of the case study is represented by a six dimensional vector. Three of its dimensions are continuous, while the other three are discrete. We propose a concept called 'Adaptive Action Set (AAS)' to limit the action set size by appropriately considering a finite number of controls. The main idea of Adaptive Action Set (AAS), is that for each state, the value function update will be restricted to only a small set of actions or controls. Our approach is similar to the *evolutionary random policy search* (ERPS) [38] method. Other than the EPRS approach and the continuous decision space representation proposed by [1], there has been little effort to address this source of the COD.

The remainder of this chapter is organized as follows. In Section 3.1, I describe in depth

the proposed RTADP approach, the concept of the adaptive action set and the explicit calculation of a single backup. Then in Section 3.2, I apply the RTADP algorithm at a manufacturing job shop under uncertain demand and product yield. This exemplary problem is formulated as a formal MDP and results are obtained during the simulation exercises. In Section 3.3, we try to identify some potential issues of the RTADP approach by applying it on 3 stochastic shortest path instances. Finally, at Section 3.4, we provide a summary for this chapter.

## *3.1 A Real-Time Approximate Dynamic Programming (RTADP) Approach*

RTDP is a variant of value iteration and asynchronous dynamic programming. It requires that the next state that one visits is determined by the current state, action and a sample realization of the exogenous information. RTDP, as is introduced by *Barto et al* [15], is combined with an optimistic estimate of the value function and assumes that the expectation operator can be calculated exactly. This extends to a stochastic setting of the A-star algorithm of AI. This initial optimistic estimate of the value function is imperative for the RTDP algorithm to guarantee convergence, but because of it the algorithm is due to visit almost the entire discrete state space during the training phase. This makes RTDP an impractical tool even for a small application. To gain some insight into the computational requirement of the RTDP, we refer the readers to *Barto et al* [15] for a numerical illustration of the RTDP approach for a stochastic shortest path problem with 9,115 discrete states. Results indicate that the value functions converge and instruct the optimal policy, when only 2% of the total states is updated more than 100 times, 20% updated more than 10 times, and 3% of the spate space not updated at all. However, 3% saving in terms of the state space is not sufficient for most practical problems. Therefore our modifications are designed so that a high quality policy can be instructed after only a tiny fraction of the state space is visited.

The RTADP approach proposed in this chapterr combines basic elements of the original RTDP approach with the $k$-Nearest Neighbor ($k$-NN) approximation scheme outlined in

[33], an underestimator that discourages exploration of previously unvisited regions of the state space and some specific ways for selecting candidate actions that comprise the AAS.

RTDP starts by performing trials, where each trial starts from a same set of starting states and accumulates information along the state trajectories generated. It is observed that after a finite number of trials the set of states that are visited during the trials becomes saturated. This set is denoted by $\mathbb{S}_{Sim}$ hereafter. As mentioned before, the relevant state space $\mathbb{S}_R$ is the set of all states that belong to the trajectories followed under the optimal policy with the same set of starting states (with a probability above some threshold in the case of stochastic systems) [15]. From a practical standpoint, there is no algorithm that can exactly identify this set of states for general problems with computation that scales polynomially with respect to the number of problem parameters. The goal is to construct $\mathbb{S}_{Sim}$ that encompasses $\mathbb{S}_R$ as much as possible.

### 3.1.1 Formal RTADP Description

The proposed episodic procedure attempts to build an evolving value table by starting with an empty one and gradually adding more and more entries, as specific states are encountered in the simulation. The following steps are involved in each trial-episode of the algorithm.

**Step 0a** Initialize a *starting* state $s_0$ as $s_t$ and $\mathbb{S}_{Sim} = \{s_0\}$.

**For episodes** $i = 1, 2, ...M$ , where $M$ is a sufficiently large integer

**For iterations** $t = 1, 2, ...h$ , where $h$ is the horizon length of each episode

**Step 1** Construct an Adaptive Action Set ($A_{sub}$) for $s_t$. $A_{sub}(s_t) \subset \mathbb{A}$, where $\mathbb{A}$ is the set of all possible controls that the decision maker can exercise at any time instance for a given state. Details about the notion of $A_{sub}(s_t)$ and how it is numerically constructed, along with an example, are given in sections 3.1.3 and 3.2.4

**Step 2** Update the value for $J^{\pi}(s_t)$ according to Eq.(12). Every control in $A_{sub}$ is evaluated with respect to the maximization operator in the Bellman equation (Eq.(12)) and the decision-maker follows a policy that is greedy with respect to the most recent estimate of the value function (Eq.(13)). This evaluation requires knowledge or an estimate

of $J^{\pi}(s_{t+1})$ for all possible successor states $s_{t+1}$ from $s_t$. Further details concerning this are outlined in section 3.1.4. Interested readers can find more information about optimality and convergence for discounted infinite horizon MDPs in [11].

$$J^{\pi}(s_t) = \max_{\alpha_t \in A_{sub}} \{ f(s_t, \alpha_t) + \gamma \sum_{s_{t+1} \in \mathbb{S}} Pr(s_{t+1}^j | s_t), \alpha_t J^{\pi}(s_{t+1}^j) \}, \gamma \in [0, 1) \tag{12}$$

$$\alpha^*(s_t) = \mathbf{arg}\max_{\alpha \in A_{sub}} \{ f(s_t, \alpha_t) + \sum_{s_{t+1} \in \mathbb{S}} Pr(s_{t+1}^j | s_t, \alpha_t) J^{\pi}(s_{t+1}^j) \} \tag{13}$$

where $s_{t+1}^j$ denotes the $j^{\text{th}}$ possible value for $s_{t+1}$.

**Step 3** A state $s_{t+1}$ is sampled from the probability distribution $Pr(s_{t+1}|s_t, \alpha^*(s_t))$, as defined by the Markov model of the random variables. Let $|\mathcal{N}_{\delta}(s_{t+1})|$ denote the number of state entries within $\delta$ distance from the sampled $s_{t+1}$. If $|\mathcal{N}_{\delta}(s_{t+1})| \geq k$ then $\mathbb{S}_{Sim} = \mathbb{S}_{Sim}$, else $\mathbb{S}_{Sim} = \mathbb{S}_{Sim} \cup s_{t+1}$. (Details about this appear at section 3.1.4). Set $t = t + 1$ and go back to Step 1.

**End**

**End**

Termination occurs when the value table saturates (with few new entries) and $\|J^{\pi}(s_{i+1}) - J^{\pi}(s_i)\|_{\infty} < \varepsilon$ , $\forall s_i \in \mathbb{S}_{Sim}$, where $\varepsilon$ is a preset tolerance parameter. Termination can also happen without convergence if a prespecified limit on the number of iterations is reached beforehand. In the latter case, it is preferable to increase M by some factor, and repeat the process, retaining the information acquired thus far.

To implement this approach, the user will need to choose a set of parameters. The user-chosen parameters required to tailor the RTADP approach to the manufacturing system example are described in the numerical results. Note that, if the algorithm happens to circulate over a small cyclic graph of states, the algorithm is reset to a new iteration.

### 3.1.2 Initialization

For best results, the initialization procedure should be tailored to the specific application. The approach will work either we initialize the $\mathbb{S}_{sim}$ with a random state or with a trajectory of states using heuristics or mathematical programming.

A general guideline to initialize the state space is to use deterministic mathematical programming by relaxing the exogenous uncertainty. In this case, the sampled states are restricted to the system states sampled along the derived deterministic mathematical programming action trace. For each state sample of that trajectory one can solve SAA mathemaitcal programs to initialize the value function.

A different way to initialize the value functions as well as the state space is via heuristics. A qualitative way to achieve that is described in [17, 18]

### 3.1.3 Key Elements of $A_{sub}$

The purpose of adopting $A_{sub}$ is to significantly reduce the COD arising from having to evaluate the value function of successor states for all candidate actions belonging to $\mathbb{A}$. The tradeoff is that with the use of $A_{sub}$ in the place of $\mathbb{A}$ we lose any formal guarantee that the quality of the value estimates for each state in the table continues to increase with the number of iterations. However, our simulation results (presented in the next section) show that the approximation quality does continue to improve with iterations.

Elements of the $A_{sub}$ include:

1. *Heuristics derived actions:* Heuristics are policies that are derived from available knowledge about the problem. These policies set a baseline performance to improve upon. "Patching up" different heuristic policies into a single policy by using the DP principle can lead to significant improvements, as illustrated in the recent literature [17] [33]. In section 3.2.4, we describe a myopic heuristic, which we use to generate actions for the studied example.

2. *Math Programming derived actions:* If we can formulate the problem of interest as a mathematical programming model, we can include those actions instructed by it. For example, the optimal action from value maximization over a finite horizon can be used. To avoid the need for solving a complicated stochastic program, a particular realization of random variables (e.g., their mean trajectories) could be used. Resulting actions, optimal for the deterministic case, represent only suboptimal actions for the original stochastic multi-stage problem. There are many variants of this approach and

it is an open question as to how much effort to invest in finding actions this way. In section 3.2.4, we formally describe the mathematical program used to generate actions for the current example.

3. *Best known actions:* These actions are derived from prior learning of the value function in the "evolving" value table. If the state to be updated is a state never visited before, then its best known action is empty. If the state has been visited before, a best known action should have been stored with respect to the prior estimate of the value function.

4. *Random actions:* Random actions allows one to explore the whole action space in the limit and hence the relevant state space. We generate limited random actions by adding random perturbations to the actions generated from the heuristics and mathematical programming.

5. *Actions associated with neighboring states:* These are the best known actions of the k-nearest neighbors of the current state.

### 3.1.4 On Calculating $J^{\pi}(s_t)$

In Eq.(12) the calculation of $J^{\pi}(s_t)$ involves the knowledge of the value function of all possible successor states $J^{\pi}(s_{t+1})$ for each action in $A_{sub}$. During the calculation, we will encounter one of the following three possible situations.

**1:** All $s_{t+1}$'s have values registered in the value table. We then use these values to calculate $J^{\pi}(s_t)$.

**2:** Some of the $s_{t+1}$'s are not found in the value table. In this case we first need to find the set of entries within $\delta$ distance of $s_{t+1}$ (to be denoted by $\mathcal{N}_{\delta}(s_{t+1})$). Here we use the weighted Euclidean distance metric $d$, as proposed by [33], with a user-chosen parameter $\delta$:

$$\mathcal{N}_{\delta}(s_{t+1}) \stackrel{\text{def}}{=} \left\{ s \in S \ : \ d = \sqrt{(s - s_{t+1})^T W (s - s_{t+1})} < \delta \right\} \tag{14}$$

In the above, $W$ is a feature weighting diagonal matrix. If $|\mathcal{N}_{\delta}(s_{t+1})| \geq k$, we approximate the value function of $s_{t+1}$ from the $k$ nearest states recorded in the value table,

by taking an average, as follows:

$$J^{\pi}(s_{t+1}) = \frac{1}{k} \sum_{x \in \mathcal{N}_k(s_{t+1})} J^{\pi}(x) \tag{15}$$

where $\mathcal{N}_k(s_{t+1})$ denotes the set containing the k nearest neighbors. In [64], the authors proved the convergence of the VI when the $k-NN$ averager is employed for the value function approximation.

In the case that $s_j$ has $k' < k = 4$ neighbor states within the specified distance, we use Eq.15 with just $k'$ states to approximate $J^{\pi}(s_j)$.

**3:** The case where $|\mathcal{N}_{\delta}(s_{t+1})| = 0$. Eq.15 cannot be used, and therefore we suggest an initial estimation scheme for the value functions with respect to the optimal one. For example, one can "under-estimate" $J^{\pi}(s_{t+1})$ of the $s_{t+1}$'s that belong to scenario 3 by using a uniform lower bound of the value function. Assuming a maximization with positive rewards, $J_t^{\pi^*}(s_{t+1})$ is positive for all states. In this case, '0' is an under-estimator for such a case. This will be referred to as the "under-estimation" scheme.

Taking this further, one may also assign negative values to parts of the state space deemed irrelevant or highly undesirable from a priori knowledge of the problem. Doing so means these states will not be explored. Such an approach is investigated in paragraph 3.2.5.

Although we do not recommend as a part of our method, we also analyze the effect of over-estimating the value function for $s_{t+1}$'s belonging to scenario 3 by using a uniform upper bound. The impact of doing so in learning and convergence will be investigated in paragraph 3.2.5.

For the purposes of this chapter, we will explore the effectiveness of the proposed approach on a specific capacity planning instance as well as on shortest path problems or GDMDP's.

## *3.2 RTADP Applied At Capacity Planning*

Capacity planning and allocation under uncertainty is one of the most complex and challenging problems faced in industrial manufacturing. In general, effective management of

such strategic decisions is crucial for the financial prosperity of an industrial firm. Typical capacity management problems involve multiple factors that can be expanded or contracted. Capacity expanding decisions are usually associated with significant investments and are often irreversible, *i.e.*, invested capital for expansion cannot be fully retrieved if one decides to contract the excess capacity later [65]. For a complete review of the investment capacity management literature, the reader is referred to [66]. For problems involving multi-factor investments under single dimensional Markovian uncertainty and linear adjustment cost structure, an analytical policy named ISD (Invest - Stay - Disinvest)[1] is derived from the optimality principle of Dynamic Programming (DP). A mathematical description of the analytical policy for such problems can be found in [67]. In the case that the stochastic process is multi-dimensional and/or the adjustment cost structure non-linear, however, the ISD policy does not guarantee optimality. For such cases, one may choose to use optimization to numerically retrieve a high quality policy. Historically, such problems have been tackled via mathematical programming techniques such as Mixed Integer Programming (MIP) [68]. Formulating and solving this type of multi-stage stochastic optimization problem using mathematical programming generally entails exploring a large number of scenarios or evaluating multi-dimensional integrals over the probability distributions, and the computational load typically increases exponentially with the number of stages [69]. As an alternative, researchers have explored stochastic Dynamic Programming (DP) methods, but these suffer from an exponential growth in computation with respect to the system's state dimension, a problem that is typically referred to as the "curse of dimensionality (COD)" [2]. No matter the approach, in the general case, there is an unavoidable tradeoff to be made between computational efficiency and solution quality for this type of problem.

The particular system of focus in this chapter is a three-stage queuing manufacturing process, illustrated in Figure 6. The objective is to control the buffers through irreversible

[1] The ISD policy partitions the space into a number of sub-regions. The *Stay* space is the region of state space, where the decision-maker chooses to maintain the same capacity for every factor. The *Invest* region is a point in the state space, where the decision maker will invest and therefore expand the capacity of one or more factors. The *Disinvest* region is a point in the state space, where the decision maker will disinvest and therefore contract the capacity of one or more factors.

capacity planning decisions and maximize the system's economic objective. One can categorize the problem as belonging simultaneously to the broad class of multi-factor investments under uncertainty and queuing network control via dynamic capacity decisions. Specifically, the manufacturing process of interest is divided into three interdependent stages with physical buffers (queues) and a final product inventory. The in-process inventory queued at each stage is controlled via the simultaneous adjustment of the capacity at each stage. The demand rate for the final product is stochastic and modeled as a first order Markov chain. A further complicating factor in this example is the possibility of processing failure: If a product coming out of stage 2 fails to meet required specifications, the failed product is rerouted to stage 3 for reprocessing. At station 3 the failed product is serviced and placed back in the queue in front of stage 2. The fraction of products that meet the specs at each time period is also a random variable, which is also modeled as a first order Markov chain. The decisions at each time consist of addition / subtraction of equipment units at each stage (at certain costs) and the number of units actually used for production during the time period. Such decision problem can naturally be mathematically formulated as a Markov Decision Process (MDP). Nonetheless, this formulation suffers from the COD, and this chapter will suggest a practical way to overcome it.

The state variable of the case study is a six dimensional vector and represents the in-process inventory at each stage and the realized values of the two random variables. All the elements of this vector are discrete-valued and we use a "flat" representation, i.e., each state is numbered 1, 2, ..., $|\mathbb{S}|$, which is in similar spirit with [17, 18, 64].

### 3.2.1 Manufacturing Job Shop Under Uncertain Demand and Product Yield

As mentioned in the introduction, we study an exemplary single product manufacturing system in which the intermediate and final product queues are controlled via capacity adjustment and utilization by a single agent. The manufacturing process is illustrated in Figure 6. A high level description of the problem was already given in the introduction section and we give more details and formulate it as a formal MDP here.

The variable $W_{i,t}$ represents the in-process inventory queued at each stage $i$ at time

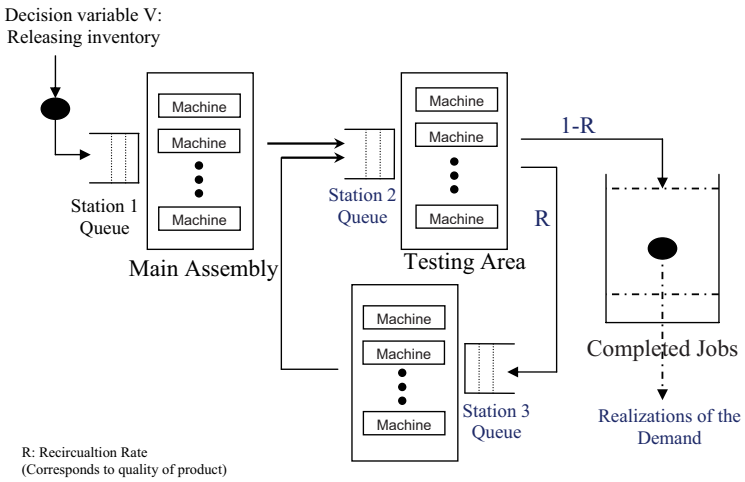



**Figure 6:** Infrastructure of an agent.

period $t$ and the variable $I_t$ the final product inventory. The demand for the final product is a sequence of random variables $D_1, D_2, D_3, ...$ with the Markov property, meaning the future values depend solely on the present state and are independent of past states. Higher-order Markov models may describe the demand pattern better in practice but this is highly case-specific and for our study we choose to go with the simplest model type. For a first order Markov chain model, the dynamics is governed by a probability transition matrix, whose $(i, j)^{\text{th}}$ element represents $Pr(D_{t+1} = d_i | D_t = d_j)$ where $d_i$ is the $i^{\text{th}}$ possible value for $D$. In this study, $i = 1, \cdots, 6$ and therefore the transition probability matrix $P_D$ is a $6 \times 6$ matrix. In practical applications, one would need to identify these probabilities mainly from historical data or reinforcements from the environment [2]. The fraction of products that fail to meet the specs at each time period is also represented by a sequence of random variables denoted by $R_1, R_2, R_3, ....$ At every time period $t$, the random variable $R_t$ may take one of two possible values. The lower state value of $r_1$ corresponds to a high product yield and conversely the higher state value of $r_2$ to a low product yield. Transition from $r_1$ to $r_2$ may be due to an unexpected event that harms the system's performance. The corresponding $2 \times 2$ probability transition matrix is denoted by $P_R$.

[2] In practice, such models can be identified using hidden Markov models and by utilizing the EM algorithm[61]

**Table 2:** Physical meaning of each term of the objective function.

| | |
|---|---|
| $\mathbf{A}_1 \min\{D_{t+1}, St_{t+1}\}$ | Revenue coming from satisfied demand |
| $\mathbf{A}_2((St_{t+1} - S_{SP})^+ + (S_{SP} - St_t)^+)$ | Penalty incurred when stock deviates from $S_{SP}$ |
| $\mathbf{A}_3(w_{2,t+1} + w_{3,t+1})$ | Holding costs at at stages 2,3 |
| $\mathbf{A}_4(\sum_{i=1}^{3} \Delta\widehat{U}_{i,t}$ | Machines available at each stage, $i = 1, 2, 3$ |

**Table 3:** Numerical values used for the manufacturing job shop example.

$$P_R = \begin{bmatrix} 0.8 & 0.2 \\ 0.6 & 0.4 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.2 & 0.7 \end{bmatrix}$$

$$P_D = \begin{bmatrix} 0.8 & 0.1 & 0.08 & 0.02 & 0 & 0 \\ 0.06 & 0.7 & 0.1 & 0.07 & 0.07 & 0 \\ 0.01 & 0.08 & 0.7 & 0.1 & 0.11 & 0 \\ 0.02 & 0.02 & 0.01 & 0.8 & 0.13 & 0.02 \\ 0 & 0.01 & 0.11 & 0.13 & 0.7 & 0.05 \\ 0.02 & 0.01 & 0.01 & 0.11 & 0.15 & 0.7 \end{bmatrix}$$

$$D = \begin{bmatrix} 7 & 20 & 35 & 50 & 70 & 100 \end{bmatrix}$$

$$\mu_1 = 10, \quad \mu_2 = 15, \quad \mu_3 = 10$$

$$A_1 = 5 * 10^{-2}, \quad A_2 = 6 * 10^{-3}, \quad A_3 = 5 * 10^{-4}$$
$$A_4 = 10^{-2}, \qquad \gamma = 0.9, \qquad S_{des} = 500$$

### 3.2.2 Formal MDP Formulation Of The Manufacturing Job SHop

The formulation of this problem as a formal MDP requires specification the following elements: State variables, exogenous information variables, decision variables, transition function, one stage profit function, and objective function. The following subsections detail each of these.

#### *3.2.2.1 State Variables / Exogenous Information Variables*

A compact definition of state given in [1] is: *a state variable is the minimal function of history that is necessary and sufficient to model all the future dynamics of the system.* For the manufacturing job shop example, the state variable is a six dimensional vector defined as below:

$$s_t = \begin{bmatrix} I_t = \text{ Finished product at time } t \\ W_{i,t} = \text{ Queue at stage } i = 1,2,3 \text{ at time } t \\ \hat{D}_t = \text{ Realized demand value at time } t \\ \hat{R}_t = \text{ Realized recirculation rate value at time } t \end{bmatrix} \tag{16}$$

For a larger queueing network, one can use the same characteristic features summarized in the above statement in defining the state variable.

As mentioned earlier, the random variables are modeled with a first order Markov model in this study. The models that govern the probabilistic transitions among the given discrete sets of values are completely specified by $P_D$ for $D$ and $P_R$ for $R$. For example, the $(i,j)^{\text{th}}$ element of $P_D$ is the probability of the demand taking the current value of $d_j$ to transition to the value of $d_i$ at the next time period. In the numerical example, the diagonal elements of $P_D$ are chosen near 1, in order to reflect the strong inertia seen in a typical demand profile.

Realized values of the random variables at time $t$, $\hat{D}_t$ and $\hat{R}_t$, are assumed to be known to the decision-maker. Hence they constitute exogenous information variables. However, the values of these variables at future times are uncertain and are described by the corresponding probability distributions. It is customary to include the parameters defining

these conditional probability distributions of the random variables as a part of the state vector as they capture the information available at that time period. They are therefore called information states. With a first order Markov chain model, it is sufficient to include just $\hat{D}_t$ and $\hat{R}_t$ in the system state as they completely define the conditional probability distributions of the future random variables.

#### *3.2.2.2 Decision Variables*

Decisions are modeled in discrete time. The decision space $\mathbb{A}$ encodes all the possible controls that are applicable to each system state $s_t$. Each action or control is concerned with: a) the capacity expansion-contraction, meaning adjustment of the number of the machines at each stage and b) the percent of the machines that are actually employed for production at each stage. Therefore $\mathbb{A}$ has six dimensions, of which the three last are continuous.

$$\mathbb{A} = \begin{bmatrix} \widehat{U}_{i,t} = \text{ Number of machines at } i \text{ at } t \\ PU_{i,t} = \text{Percentage of } \widehat{U}_{i,t} \text{ used at } t \end{bmatrix} \tag{17}$$

#### *3.2.2.3 Transition Function*

The total number of jobs $TJ_{i,t}$ coming out from station $i$ at time $t$ is expressed via Eq.(18):

$$TJ_{i,t} = \mu_i PU_{i,t} \widehat{U}_{i,t} \quad \forall t (i = 1, 2, 3) \tag{18}$$

where $\mu_i$ is the capacity of each machine at stage $i$, which is assumed to be constant.

The key element in modeling the system dynamics is to take into account the physical constraints (*i.e.*, material balances) among the interdependent and sequential stages. Considering stage 1 to be the *main assembly area* of the product, we can balance the inlet and outlet flows as shown in Eq. (19):

$$W_{1,t+1} = W_{1,t} + V_t - TJ_{1,t} \tag{19}$$

$$W_{1,t} \geq 0, \quad \forall t$$

where $V_t$ is the number of raw ("beginning") products coming into the system.

The next stage is the *testing area* where the product is tested extensively to ensure that the quality requirements are met. The recirculation rate $R$ takes one of the two values at each time period according to the realization of the Markov chain with the transition matrix of $P_R$ (see Table 3). Accordingly, the material balance for the queue at stage 2 is shown in Eq. (20).

$$W_{2,t+1} = W_{2,t} + TJ_{1,t} + TJ_{3,t} - TJ_{2,t} \tag{20}$$

$$W_{2,t} \geq 0, \quad \forall t$$

A fraction of the product, $R$, fails the quality test and needs to be reprocessed at station 3. After being reprocessed the product will have a chance to pass the quality test again. It is assumed that the product has no memory of whether it has passed or failed the test previously.

The material balance for the queue at stage 3 is shown in Eq. (21)

$$W_{3,t+1} = W_{3,t} + R_t TJ_{2,t} - TJ_{3,t} \tag{21}$$

$$W_{3,t} \geq 0, \quad \forall t$$

The inventory of finished products after satisfying the demand at each time period is described by (22).

$$I_{t+1} = I_t - D_t + (1 - R_t)TJ_{2,t} \tag{22}$$

$$I_t \geq 0, \quad \forall t$$

Since the stochastic processes of the demand and recirculation are independent, the transition to the next demand and recirculation rate follows the conditional probability distributions $P_D$,$P_R$ and hence is dictated by the stochastic outcome of a corresponding biased coin. The possible outcomes for the demand rate are 6, while for the recirculation rate are 2 (the numerical details appear at Table 3).

#### *3.2.2.4 Contribution (Profit) Function*

The one step profit produced by a decision $\alpha_t$ at state $s_t$ during one time period with random variable $\omega_{t+1}$ is denoted as $\hat{f}(s_t, \alpha_t, \omega_{t+1})$. With some abuse of notation we denote

$\omega_{t+1} = [D_{t+1}, R_{t+1}]$, which is a two dimensional vector describing the outcome of the two independent Markovian processes at time $t+1$. Then, the expression for $\hat{f}(s_t, a_t, \omega_{t+1})$ is:

$$\begin{aligned}\hat{f}(s_t, a_t, \omega_{t+1}) = \mathbf{A}_1 \min\{D_{t+1}, I_{t+1}\} - \mathbf{A}_2((I_{t+1} - S_{SP})^+ + (S_{SP} - I_{t+1})^+) \\ -\mathbf{A}_3(W_{2,t+1} + W_{3,t+1}) - \mathbf{A}_4(\textstyle\sum_{i=1}^{3} \widehat{U}_{i,t})\end{aligned} \tag{23}$$

In the above, the notation $\{\cdot\}^+$ is defined as $y^+ = max(y, 0)$ assuming $y$ is a real number. The physical interpretation of the terms in Eq.(23) and the meaning of the parameters $\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3, \mathbf{A}_4$ are given in Table 2. The numerical values of the solved instance are summarized in Table 3. A high quality policy is one that achieves a good balance among the objectives of a) controlling the stock level $I_t$ close to a fixed set point $S_{SP}$, b) minimizing the queues $(W_{2,t}, W_{3,t})$ at stations 2 and 3, and c) minimizing the resources used.

The expectation of the one step profit is defined over the probability space $\Omega$:

$$f(s_t, a_t) = \mathbb{E}[\hat{f}(s_t, a_t, \omega_{t+1}|s_t)] = \sum_{j=1}^{N} P_j(t+1)\hat{f}_j(s_t, a_t, \omega_j) \tag{24}$$

where $P_j(t+1)$ is the probability of $\omega_{t+1}$ taking the value of $\omega_j$ and $N$ is the number of transitions with non-zero probability starting from $s_t$.

#### *3.2.2.5 Objective Function*

The primary objective in this problem is to find the policy $\pi$ that maximizes the discounted expected profit over infinite horizon.

$$\max_{\pi} \left\{ J^{\pi}(s_0) = \mathbb{E}\{ \sum_{t=0}^{\infty} \gamma^t f(s_t, \pi(s_t)) \mid s_0 \} \right\} \tag{25}$$

This goal will be accomplished when we construct a stationary decision function $\pi : s_t \rightarrow a_t$ such that each state is mapped to the best possible action.

Trivially, the optimal policy $\pi^*$ is the one instructed by the optimal value function. The RTADP methodology discussed next can be applied to generate policies that are generally suboptimal but are conjectured to be close to the optimal one.

### 3.2.3 Simulation Results

The results of the presented manufacturing job shop capacity adjustment problem are organized as follows: In section 5.1, we describe step-by-step a) the simulation procedure, b)

how $A_{sub}$ is numerically constructed, and c) the parameters used to characterize $\mathcal{N}_\delta(s_{t+1})$. In section 5.2 we evaluate the performance of the proposed RTADP method with a heuristic policy derived from problem insights, an ideal (but not practically implementable) solution derived from Mixed Integer Programming (MIP), and an implementable solution based on rolling horizon MIP. The "ideal" solution is not implementable as it is based on solving an MIP assuming *full information* about future realizations of the uncertain parameters. However, it does provide an absolute upper-bound to compare against, and demonstrates how the knowledge of future impacts the performance.

### 3.2.4 Simulation Procedure

The simulation begins with an empty value table. In simulating the problem, the following assumptions are made: a) No machine breakdowns , b) identical machines used for each stage, c) two machines in each station at minimum (*i.e.*, $\min\{\widehat{U}_{i,t}\} = 2$), d) negligible holding costs, order costs, and quantity discounts for the raw materials released into the initial queue, e) sufficient raw material availability to allow for a desired level of processing ($\mu_1 PU_{i,t}\widehat{U}_{i,t}$) at all times, f) fixed selling price of the product as well as the operation expenses (e.g., energy consumption, maintenance fee) over the time horizon of interest, and g) responsive delivery of the raw materials with no delay.

Next we discuss the construction of $A_{sub}$. Recall that $A_{sub}$ is composed of several elements, which are constructed for the problem of interest in the following manner.

**Heuristic action:** The heuristic we adopt works in the following fashion. At $t$, the agent receives the state information. Upon observing the realization of the random variables ($D_t$ and $R_t$), one-step-ahead predictions about the random variables are made, which we denote by $D_{t+1|t}$, $R_{t+1|t}$, by maximizing the conditional probability. This can be done easily since we have assumed perfect knowledge of the one step probability transition matrices. The heuristic is myopic in that it sets to achieve the following regulatory objective at the next time step: The one-step-ahead prediction of $I_{t+1}$ should be at its setpoint, $S_{SP}$ (i.e., $I_{t+1|t} = S_{SP}$ ), and the queue for the $2^{nd}$ and that for $3^{rd}$ stage are zeroed ($W_{3,t+1} = 0$ and $W_{2,t+1} = 0$). Let us introduce the effective number of machines $\widetilde{U}_{i,t}$, which is defined

as $\widetilde{U}_{i,t} = PU_{i,t} \times \widehat{U}_{i,t}$. Using the one-step-ahead predictions $D_{t+1|t}$ and $R_{t+1|t}$, the effective number of machines in operation at station 2 (Eq. 26), station 3 (Eq.27), and station 1 (Eq. 28) are adjusted to meet the queue level objectives, *i.e.*, $\widetilde{U}_{i,t+1}$ values are chosen according to

$$\widetilde{U}_{2,t+1} = \frac{S_{SP} - I_t + D_{t+1|t}}{\mu_2(1 - R_{t+1|t})} \tag{26}$$

$$\widetilde{U}_{3,t+1} = \frac{W_{3,t} - 0 + \mu_2 r_{t+1|t} \widetilde{U}_{2,t} D_{t+1|t}}{\mu_3(1 - R_{t+1|t})} \tag{27}$$

$$\widetilde{U}_{1,t} = \frac{0 - W_{2,t} - \mu_3 \widetilde{U}_{3,t} + \mu_2 \widetilde{U}_{2,t}}{\mu_1} \tag{28}$$

$\widehat{U}_{i,t}$ should be bigger than $\widetilde{U}_{i,t}$ and must have an integer value. Therefore this heuristic chooses $\widehat{U}_{i,t}$ as the minimum integer that satisfies these properties. Also, when $I_t > S_{SP}$, then the shop operates with minimum resources (see assumptions).

**Mathematical Programming Actions:**

$A_{sub}$ includes actions derived from math programming, which is an MIP. The decision variables for this formulation are $\widehat{U}_{i,t}$, $\widetilde{U}_{i,t}$ $\forall$ $i = 1, 2, 3$ and $t = 1, ...h$ (where $h$ is the length of the horizon of the MIP). The formulation, which requires that their realized stock is $St \leq S_{SP}$, is:

$$\begin{aligned}
&\max \textstyle\sum_{t=1}^{h} \Big( f(s_t, \widetilde{U}_{i,t}, \widehat{U}_{i,t}) \Big) \\
&s.t. \\
&g(s_t, \widehat{U}_{i,t}, \widehat{U}_{i,t}) = 0 \\
&-\epsilon_1 \leq \Delta \widehat{U}_i(t) \leq \epsilon_1 \\
&PU_{i,t} \geq \beta \\
&I_t \leq S_{SP} \\
&W_{i,t} \geq 0, \widehat{U}_{i,t} \geq 2
\end{aligned} \tag{29}$$

$g(s_t, \widetilde{U}_{i,t}, \widehat{U}_{i,t})$ denotes the material balances that must be satisfied at each system node for consecutive time periods. The incremental change, $\widehat{U}_{i,t+1} - \widehat{U}_{i,t}$ is denoted as $\Delta \widehat{U}_{i,t}$, which cannot be greater or less than $\epsilon_1$. Parameter $\beta$ is a parameter corresponding to the minimum percentage of the available machines that must be used at each time period. At each iteration of the RTADP, we systematically used 7 distinct MIP, by tuning the $\beta$ values at (0.4,0.5,0.6,0.7,0.8,0.9,1) and $\epsilon_1 = 6$. The action at $t = 1$ is added to $A_{sub}$.

If the realization of the state $s_t$ is above $S_{SP}$, then the MIP becomes infeasible. Specifically, if $I_t > S_{SP}$, we utilize the $I_t > S_{SP}$ constraint instead of $I_t \leq S_{SP}$ in order to ensure the feasibility and produce numerical actions from the MIP code.

To complete the $A_{sub}$ at each loop, we also include: four distinct random actions, a best known action among stored actions if there is any, and the best stored actions from the $k$-nearest neighbors of $s_i$ (where $k = 4$). The random actions are generated by adding perturbations to the heuristic and mathematical actions. The neighborhood of $s_t$ is defined by the tuning parameters $\delta$ and $W$. The Euclidean measure $\delta$ demarcates the "neighborhood" of a particular $s_t$. The states that are recorded in the evolving value function table within the so defined neighborhood are neighboring states. We choose $\delta = 12.2$ and $W$ to be a diagonal matrix with entries $W(1,1) = 0.15, W(2,2) = 0.1, W(3,3) = 0.15, W(4,4) = 4, W(5,5) = 2000$. These weights are chosen such that only those states with the same realization of the random variables and similar ($\pm 10$ jobs in queue 2,3) queue lengths belong to a same neighborhood. The value $k$=4 was used for our numerical results.

### 3.2.5 Performance Comparisons

We investigate the performance of the proposed RTADP against 1) a MIP with full information 2) a myopic heuristics-based policy and 3) a solution obtained via a rolling horizon MIP. As mentioned in section 3.2.4, there exist several options for estimating the value function of previously unseen states. The following subsections delve into this aspect further. RTADP-$a$, RTADP-$b$ and RTADP-$c$ correspond to the above three policies and are discussed in sections 3.2.5.

#### *3.2.5.1 RTADP with an Under-Estimator (RTADP-a)*

Because this is a maximization problem with positive rewards, $J^{\pi^*}(s)$ is uniformly positive. Therefore, the value '0' is a uniform under-estimator for all states. Under scenario 3 in section 3.1.4 ( $|\mathcal{N}_\delta(s_{t+1})| = 0$), we use '0' as the value of $s_{t+1}$. We also set $h = 100$ for solved mathematical programs.

Given this scheme, RTADP successfully accumulates valuable learning of the relevant state space and its associated value function, to yield a policy that optimizes the state

trajectory over a recurrent class of states. This argument is supported by Fig. 2, where it is obvious that the exploration of the state space is significant only for the first 5,000 RTADP value function updates (2,670 states collected). After that the frequency of encountering new states is reduced significantly. At 100,000 updates only 8,165 states have been encountered. New states are scarcely encountered from that point on. This implies that the 'evolving' state space has become 'saturated.' This recurrent class of states is thought to 'cover' the relevant state space as it controls $I_t$ close to $S_{SP}$ and minimizes the queues at the second and third stage. The use of the under-estimator allows for cautious and selective exploration of the state space, and gives a good compromise between exploration and computational feasibility.

Assuming ergodicity, one way to evaluate the performance of a control policy in a stochastic application is to simulate the closed-loop system over a sufficiently long horizon. Table 4 summarizes the average performance per time period that the tested algorithms achieve. Note that the MIP with full information cannot be solved for a long horizon due to the exponential increase of the problem size. Instead, we solved a full information MIP for 100 different scenarios, representing different realizations. Each scenario length was 300 time steps. Reported are the average performance values over the 100 scenarios. The MIP formulation requires knowledge of the realized values of the problem's random variables. Hence the MIP solution represents an idealistic policy that cannot be implemented in practice. The comparison between the MIP and RTADP shows the performance gap of $\simeq$14% in favor of the MIP. The performance gap is due to a combination of the full information assumption and sub-optimality of the RTADP-a.

Another issue to explore is the importance of considering the multistage nature of this problem. This question can be partially answered by comparing a myopic 1-step ahead heuristic procedure with the RTADP. Note that the 1-step-ahead heuristic policy utilizes the current information of the random variables and 1-step-ahead predictions about them. Its performance was $\sim$27% worse than RTADP-a. RTADP-a therefore gave significant improvement upon the performance of this simple heuristic. This confirms the fact that ADP approaches are more suitable to treat multistage decision problems than myopic heuristics.

**Table 4:** Evaluating the average performance per time period starting from $s_t$ = [100 100 413 20 0.2] of a)MIP with full information b) RTADP- Scheme a,b,c c) 1-Step heuristic and d)Rolling horizon MIP ($h = 60$ and $k = 1$)

| Comparison | Average Performance per Iteration | Performance Gap |
|---|---|---|
| MIP | 2.54 ± 0.56 | - |
| RTADP - *a* | 2.18 ± 0.71 | $\simeq$ 14.2% |
| RTADP -*b* | 1.97 ± 0.92 | $\simeq$ 22.4% |
| RTADP -*c* | 1.57 ± 0.98 | $\simeq$ 38% |
| 1-Step Ahead Heuristic | 1.59 ± 0.96 | $\simeq$37.4 % |
| Rolling Horizon MIP | 2.01 ± 0.68 | $\simeq$20.88 % |

A general procedure that can address multi-stage stochastic decision problems is a rolling horizon MIP approach with a sampled future scenario. In this example an average of a given number of scenarios, chosen based on the conditional probability distribution of the future values for demand was used. The results of the rolling horizon MIP are summarized in Table 5. The adjustable parameters of the rolling horizon approach are: a) the horizon length $h$ and b) the frequency at which the MIP will be solved (once every $k$ steps). The solution strategy is: *a*)given the system state, generate 100 independent sample scenarios conditioned on the current realized values of the problem's random variables and average them, *b*) solve the MIP for horizon $h$ based on the averaged trajectories and implement the MIP's solution for the first $k$ steps, and *c* repeat *a* and *b* at the next time step defined by $k$. Table 5 confirms that the quality of the rolling horizon policy is increased if we solve the MIP at every time step (after observing the realized values of the random variables). Also, the performance of the rolling horizon approach levels out for $h > 60$ The policy obtained by applying the RTADP-a is on average 7.8% better than the one yielded by the rolling horizon approach.

#### *3.2.5.2 RTADP with Prior Knowledge (RTADP-b)*

One may attempt to exploit prior knowledge about "irrelevant" portions of the state space in the following manner: To every state $s_{t+1}$ deemed "irrelevant", a 'discouraging' initial value is assigned. For a maximization problem with positive rewards, the values for such

**Table 5:** Average performance per period of the policy as derived from rolling horizon MIP approach with a given Horizon $h$ solved per $k$ time periods.

| MIP Horizon | MIP Solved per $k$ periods | | |
|---|---|---|---|
| | $k$=1 | $k$=2 | $k$=3 |
| $h$ =40 | 1.889 | 1.8516 | 1.7818 |
| $h$ =50 | 1.986 | 1.9128 | 1.8039 |
| $h$ =60 | 2.0144 | 1.9142 | 1.8239 |
| $h$ =65 | 2.0047 | 1.9325 | 1.8553 |
| $h$ =70 | 2.0085 | 1.9414 | 1.8591 |

'unwanted' or irrelevant states are assigned to be negative.

In this problem, the undesirable states are those with 'large' inventories. To implement this, those states with values of $W_i$ greater than a threshold $\theta_i$, which is a tuning parameter, were penalized. In the numerical example, we chose $\theta_i = 600, \forall i$ . Recall that the mechanism to impose an exploration barrier is to assume a low value for these states. We assigned a negative value of '-10' to those states, and ran the RTADP with the same seed for the random variables as in the previous case. We termed this run RTADP-*b*.

Such initialization will reject the actions that will take the state trajectory to the unwanted parts of the state space. The simulation results were the following: a) the cardinality of the evolving approximation of the relevant state space was $|\mathbb{S}| = 2,700$, and b) the average performance per time period was decreased to 1.97 compared to 2.18 that was previously achieved. This demonstrates the difficulty and risk associated with using such prior knowledge: Seemingly reasonable choices can negatively impact the performance.

#### *3.2.5.3 RTADP with an Over-Estimator (RTADP-c)*

There is some numerical evidence and theoretical validation for the idea that introducing optimism in the face of uncertainty leads to $\epsilon$ optimal behavior in zero sum games and MDPs. Readers are referred to *Brafman and Tennenholtz* [70] for references concerning this matter. As in their approach, our algorithm introduces no initial bias with respect to the selection of the optimal action. In the spirit of *Kearns et al* [71], our approach tries to empirically identify an irreducible set of high quality states. It is also aligned with the

R-max algorithm in that it will pick the greedy controls.

RTADP-$c$ represents the proposed approach combined with an over-estimator. This type of initialization will encourage a maximum level of exploration of the state space. The details of the over-estimation scheme are presented in Appendix A. An important characteristic of this scheme is that it over-estimates the value function for all unvisited states. Therefore, for the same random seed, RTADP-$c$ liberally explores $\mathbb{S}$, without focusing the calculation to previously visited states. This behavior is displayed in Fig.2. The state space is continually explored due to the fact that we have random actions in $A_{sub}$. If we overestimate the true value function, the RTADP-c falsely believes that every unvisited portions of the state space is a good candidate for the next state. A logical conclusion is that if RTADP is combined with such an initialization scheme, it will take a long time to converge to a high quality policy, since it will first have to explore just about the entire $\mathbb{S}$. The performance of the policy derived from RTADP-c is inferior to that from RTADP-$a$ and $b$, given the computational limitations of the current study.

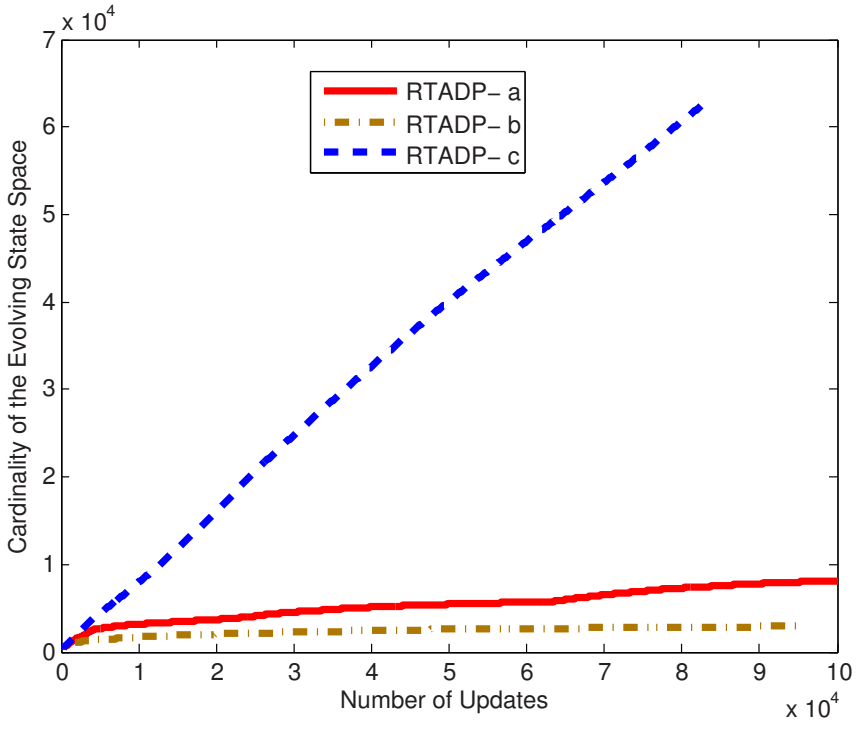



**Figure 7:** Exploration of the 'evolving' state space .

In conclusion, the initial estimation scheme used in RTADP-$a$ is the most effective in

balancing exploration and exploitation for the specific case at hand.

#### *3.2.5.4 Finished Stock - Behavior under RTADP*

This section investigates how the four aforementioned policies controls the final product inventory. The simulation results show in in Fig. 3 are for a) RTADP-$a$ b) 1-Step Ahead Heuristic c) Rolling Horizon MIP, and d) MIP with full information.

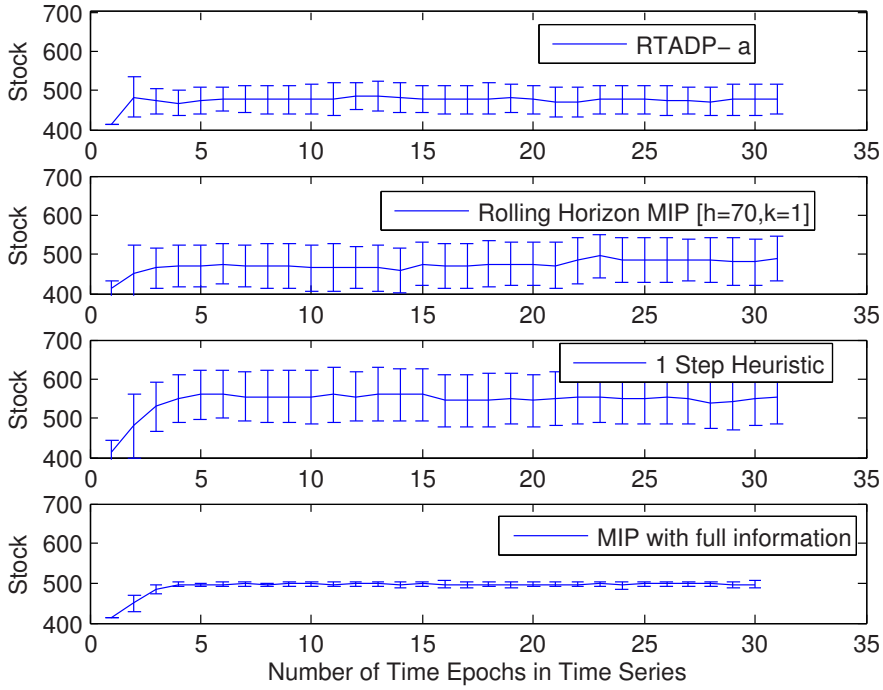



**Figure 8:** Error bounds concerning the stock level control in time series for 100 different scenarios, using the following architectures a) RTADP- $a$ b) Rolling horizon MIP c) 1- Step Ahead Heuristic d) MIP with full information .

100 different scenarios, each of horizon 30, were generated using the Markov chain model. The mean and variance of the stock level at each time point were then calculated under each of the control policies. A larger number of scenarios (1000) were run for the RTADP-$a$ approach to check that the mean and variance values had indeed converged. The mean of each discrete point of the time series for RTADP- $a$ is very close to $S_{SP}$. The error bar shown in Fig. 3 represents one standard deviation and is smaller for RTADP-$a$ than those for the heuristic and the rolling horizon MIP. The MIP with full information has the knowledge of the values of the random variables for each scenario, prior to action selection,

and therefore it controls the $I_t$ to $S_{SP}$ perfectly with minimal standard deviation.

## *3.3 Applying The RTADP Algorithm On Stochastic Shortest Path Instances - Exploring Potential Issues*

The traditional RTDP algorithm was initially applied at shortest path instances with dedicated starting and goal states. In this section, I will use instances from the special class of problems termed Goal Directed Markov Decision Processes (GDMDP's) or stochastic shortest path problems to demonstrate potential issues the user may face when he/she uses RTADP. The formulation of those instances as MDP's is shown at the previous chapter at paragraph 2.7.

In order for the traditional RTDP to achieve the optimal policy, the user needs to assign an optimistic value function approximator for all the states not yet explored within the state space. The intuition behind this fact is that, the RTDP procedure will pursue constant exploration to the unexplored regions of the state space. Therefore, it will explore almost the entire state space and after exploration is finalized, then it will focus the exploitation and on converging the value function values. This explains the fact that for large scale discrete event stochastic problems, when using such an optimistic approximator for all the unseen successive states the RTDP routine is computationally intractable.

Therefore, for the scope of this thesis, I suggested the usage of the RTADP methodology, in order to achieve computationally tractability by restricting in a heuristic way the exploration. Such alsgorithm sacrifices the formal convergence property to the optimal value function, since it will not explore the entire state space, We also further "destroyed" ,in the sake of computational feasibility, the contraction property of the DP operator by introducing the concept of the adaptive action set.

Next, I consider 3 stochastic shortest path instances. Each application differs in the cost structure and by one order with respect to the cardinality of the state space. The intent is to demonstrate some practical issues with respect to the RTADP approach and how one can optimize its performance.

The examined issues are the following:

- How will the usage of a pessimistic versus an optimistic value function approximation scheme for all unseen states affects the achieved value table expansion and the quality of the constructed policy ?
- How will a potential state space initialization scheme along with a pessimistic or optimistic value function approximation scheme help the approach to yield better policies ?
- What is the significance of a heuristic policy applied to the states outside the sampled ones to the overall achieved performance gained by the RTADP ?

To provide quantitative answers to such questions, I test the following RTADP variances to the stochastic shortest path instances:

- RTADP Variance 1: RTADP + State Space Initialization Via Relaxed LP + Pessimistic Estimation Scheme + heuristic 1.
- RTADP Variance 2: RTADP + State Space Initialization Via Relaxed LP + Optimistic Estimation Scheme + heuristic 1.
- RTADP Variance 3: RTADP + No State Space Initialization + Pessimistic Estimation Scheme + heuristic 1.
- RTADP Variance 4: RTADP + No State Space Initialization + Optimistic Estimation Scheme + heuristic 1 .
- RTADP Variance 5: RTADP + No State Space Initialization + Pessimistic Estimation Scheme + heuristic 2.

Across all the stochastic shortest path instances: 1) the pessimistic value function estimation, when encountering a state outside the $\mathbb{S}_{sim}$ is set to 550 , 2) we utilize the $k$-NN non parametric approximation scheme, 3) the optimistic value function initialization, when encountering states outside the $\mathbb{S}_{sim}$ is set to 0 , 4) the state space initialization scheme is achieved via a deterministic LP (details about the LP formulation can be retrieved in Appenxix A).

### 3.3.1 Results On A 77 Discrete State Space Example

The first instance appears at Fig.9 along with its cost structure.

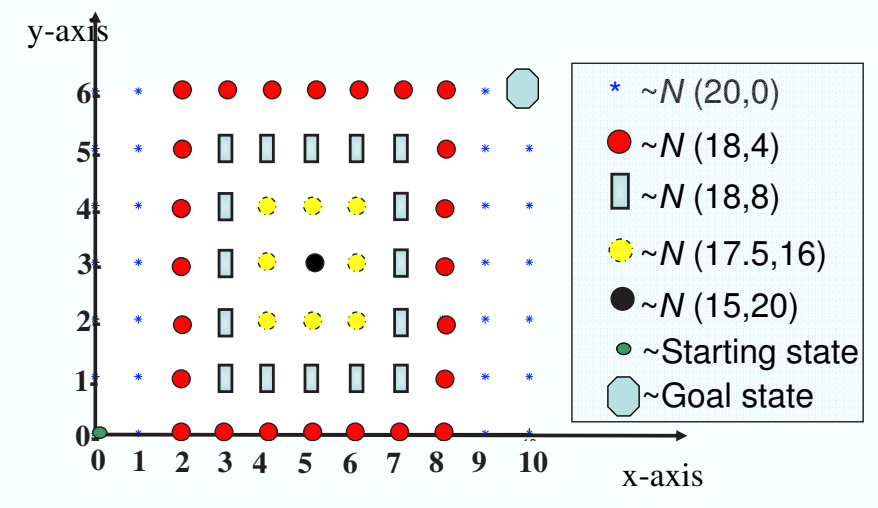



**Figure 9:** Schematic illustration of the cost structure of a Stochastic Shortest Path With 77 Discrete States.

In each state the four compass directions of movement are possible, which cause the agent to move with probability **p** in the corresponding direction on the grid and with probability (1-**p**)/ 3 in a different direction. Actions that would take the agent off the grid leave its location unchanged and they result in no penalty. The following results for this instance correspond to parameter **p** set to 0.9.

The solution via full DP appear at Fig.10.

The full DP solution for this problem yields a policy which is associated with a value function of 309 for the starting state (1,1). Below I apply the variances of the RTADP for the same instance. To derive the performance of each variance, I test them on the same 1,000 scenarios. Each RTADP variance is applied for only 1,000 episodes each of which was of length 50 simulated time steps.

For this instance the deterministic LP provides correct actions for the states involved to the deterministic trajectory from the starting to the goal state. Thats no great surprise given that $\mathbf{p} = 0.9$.

For RTADP Variances 1,2,3,4 the heuristic 1 completes the policy, since these variances

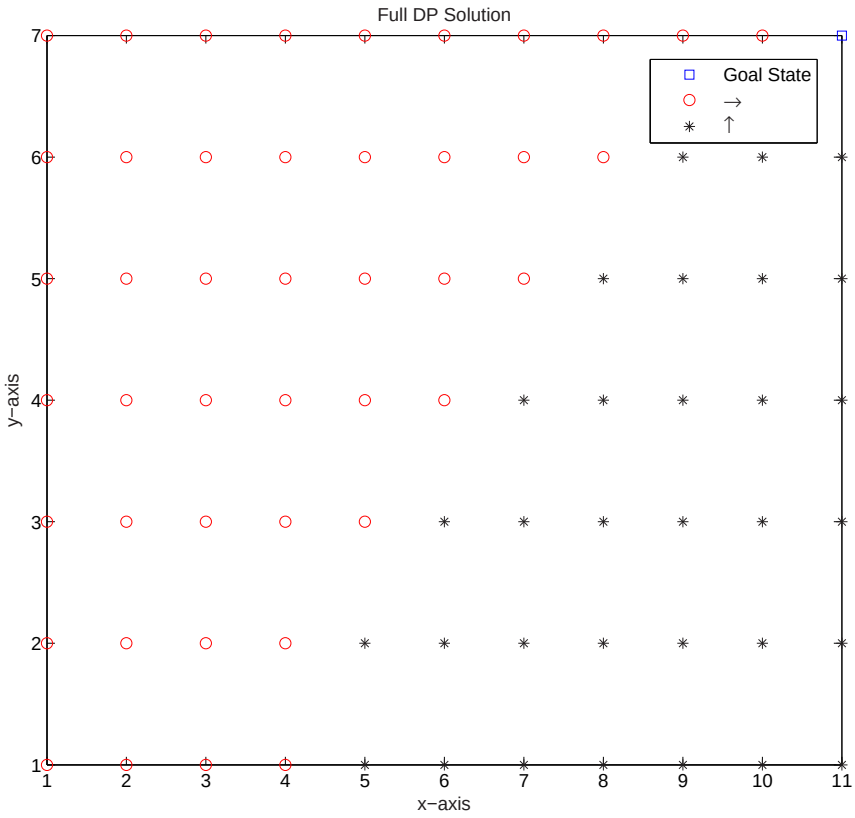



**Figure 10:** Full Dynamic Programming result for the problem as it appears at Figure 8.

essentially identify a portion of the state and therefore a partial policy. When evaluating the value function of that policy, in the case that the system goes to an unregistered to the table state heuristic 1 instructs the action.

Heuristic 1 instructs to use for all states the go up action except for the states:$\{(1,7),(2,7)$ $,(3,7),(4,7),(5,7),(6,7))\}$, where we should use the go right action . Similarly heuristic 2 instructs to use for all states the go up action except for the states:$\{(1,7),(2,7,(3,7),(4,7),$ $(5,7),(6,7))\}$, where we shall use the go down action.

The results that are summarized at Table 6 are rather inconclusive and that has to do mainly because of the size of the problem. Note that via these results we want to determine: 1) Would an effective initialization scheme matter or not, 2) Should we use a pessimistic or an optimistic estimation for the value functions of states not registered at the value table,

**Table 6:** Comparison between the performance gained from RTADP variances and full dynamic programming on the stochastic shortest path as it appears at Figure 8.

| Algorithm | Online Performance | States Explored (Percentage of space explored) |
|---|---|---|
| Full DP | 309 | Entire Space |
| RTADP Variance 1 | 309.31 | 51/77 (66.23%) |
| RTADP Variance 2 | 309.65 | 59/77 (76.62%) |
| RTADP Variance 3 | 311.61 | 62/77 (80.52%) |
| RTADP Variance 4 | 309.75 | 64/77 (83.12%) |
| RTADP Variance 5 | 321.39 | 62/77 (80.52%) |

3) how much will a heuristic that would complete the policy affect the performance, is this element critical for our approach.

What follows is to apply the RTADP to a larger instance of 900 discrete states.

### 3.3.2 Results On A 900 Discrete State Space Example

The purpose of this paragraph is to demonstrate the discussed RTADP variances to a 900 discrete state space shortest path problem in order to derive useful conclusions about the characteristics that enhance RTADP's performance.

The following results for this instance correspond to parameter **p** set to 0.8. The full DP solution for this problem yields a policy which is associated with a value function of 237 for the starting state (1,1). The cumulative results for the RTADP variances appear at Table 7.

Heuristic 1 instructs to use for all states the go up action except for the states:$\{(1,30)$, $(2,60),...,(28,840),(29,870))\}$, where we should use the go right action . Similarly, heuristic 2 instructs to use for all states the go up action except for the states:$\{(1,30),(2,60),...,(28,840)$ $,(29,870))\}$, where we shall use the go down action.

**Table 7:** Comparison between the performance gained from RTADP variances and full dynamic programming on the stochastic shortest path with 900 states.

| Algorithm | Online Performance (Relative to Full DP Percentage) | States Explored (Percentage of space explored) |
|---|---|---|
| Full DP | 237 | Entire state space |
| RTADP Variance 1 | 246.35 | 311/900 (34.56%) |
| RTADP Variance 2 | 253.46 (85.13%) | 790/900 (87.78%) |
| RTADP Variance 3 | 247.43 (83.27%) | 316/900 (35.11%) |
| RTADP Variance 4 | 285.21 (78.63%) | 840/900 (93.33%) |
| RTADP Variance 5 | 251.83 | 323/900 (35.89%) |

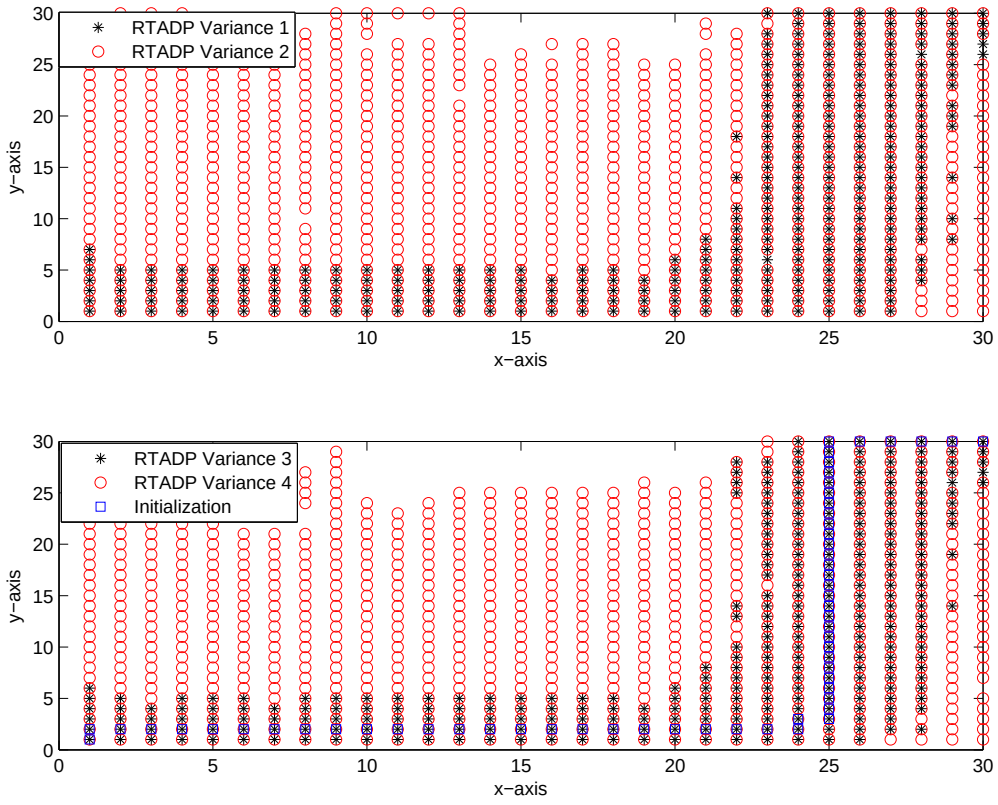



**Figure 11:** Schematic illustration of the exploration achieved by the proposed RTADP variances. Its evident that using the right type of approximator for the unseen states will result to restrict the exploration and enhance computational feasibility.

The results are summarized at Table 7. The results are pointing to the fact that if RTADP is used along with a pessimistic estimation scheme, we achieve restricted exploration and then we focus on exploitation. The derived policy from RTADP Variance 1 is very close to the optimal policy and surprisingly we are exploring only one third of the state space. If an optimistic estimation is used (Variance 2) along the RTADP then we will have to explore the entire state space and then we will asympotically converge. This is not viable in large scale problems.

Moreover, in the question of should we use an initialization scheme or a better heuristic is indifferent given these results.

What follows is to apply the RTADP to a larger by one order of magnitude state space of 10,000 discrete states.

### 3.3.3 Results On A 10,000 Discrete State Space Example

The purpose of this paragraph is to demonstrate the discussed RTADP variances to a 10,000 discrete state space shortest path problem in order to derive useful conclusions about the characteristics that enhance RTADP's performance.

The following results for this instance correspond to parameter **p** set to 0.8. The full DP solution for this problem yields a policy which is associated with a value function of 519 for the starting state (1,1). The cumulative results for the variances appear at Table 8.

Heuristic 1 instructs to use for all states the go up action except for the states $:\{(1,100),(2,200),...,(98,9800),(99,9900))\}$, where we should use the go right action .

The results are summarized at Table 8. The results are similar with the previous case and are pointing to the fact that if RTADP is used along with a pessimistic estimation scheme, the exploration is minimized and directed to the state space where high quality solutions lie. At Fig.12 we can notice the vast difference in the exploration that is guided by the different RTADP variances.

**Table 8:** Schematic illustration of the exploration achieved by the proposed RTADP variances. Moreover, the initial trajectory, from the start to the goal state, generated by the LP is well represented at those figures .

| Algorithm | Online Performance (Relative to Full DP Percentage) | States Explored (Percentage of space explored) |
|---|---|---|
| Full DP | 519 | Entire State Space |
| RTADP Variance 1 | 945.94 | 1,909/10,000 (19.09%) |
| RTADP Variance 2 | 1,068.47 (85.13%) | 7,345/10,000 (73.45%) |
| RTADP Variance 3 | 960.52 (83.27%) | 1,883/10,000 (18.83%) |
| RTADP Variance 4 | 1,176.5 (78.63%) | 7,032/10,000 (70.32%) |

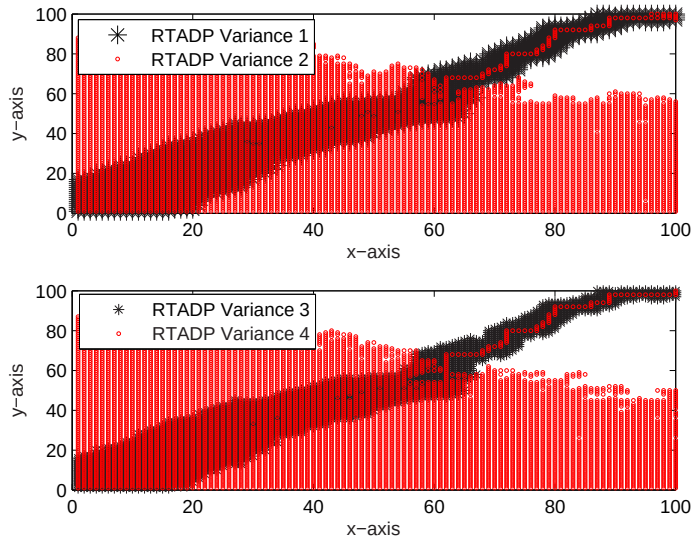



**Figure 12:** Comparison between the exploration achieved by the RTADP variances. Its evident that using the right type of approximator for the unseen states will result to restrict the exploration and enhance computational feasibility.

## *3.4 Chapter Conclusions*

For large scale problems, approximate dynamic programming has emerged as an effective way to approximate the conceptually elegant but computationally inefficient dynamic programming algorithm. The compromises made in ADP result in a tradeoff between the exploration of the state space and the exploitation of existing knowledge of the values of the already-visited states. The balance of exploitation and exploration is governed by the subset of actions allowed in each state and the relative optimism embedded in the initial assigned values to unexplored states.

In this chapter, we explored these issues in the context of a manufacturing system with variable capacity and inventory decisions faced with uncertainty over the demand and performance of the system. In this RTADP approach, a reduced action candidate set called adaptive action set was used. Its function is to control the balance between random actions for exploration and actions that were known to be good for the encountered state based on limited value function updates and established heuristics. The initialization of the state values also plays a significant role in the performance of the RTADP method. The more optimistic the initial valuation of unexplored states, the more exploration and therefore the higher the computational cost. In the example, the most effective scheme was the one that is 'neutrally pessimistic' and not biased against any particular class of states. This was effective in this problem because there was relatively good information on actions based on heuristic evaluation of their immediate consequences. In problems where such knowledge does not exist, a higher level of exploration may be required for effective actions to be discovered.

# CHAPTER 4

# SOLVING A HIGH DIMENSIONAL LIGHT AROMATIC SUPPLY CHAIN EXAMPLE USING RTADP

In this chapter, we implement the RTADP methodology for solving a high dimensional supply chain network, which is to be optimized via dynamic decisions.

The specific application is the light aromatic , or else called BTX, supply chain network. The uncertainty lies in the demand and price, whose variation is modeled using first order Markov Chains. The main decisions are the mode and the operation of the equipment as well as the stream flows. The resulting Markov Decision Process is addressed via the RTADP approach.

The proposed RTADP method starts with numerical actions derived from a Mixed Integer Linear Programming (MILP) formulation and gradually learns a superior quality solution by interacting with the stochastic system via simulation. The performance of the learned solution is evaluated against an 'ideal' solution derived using a Mixed Integer Linear Programming (MILP) formulation, which assumes full knowledge of future realized values of the stochastic variables, and a sample path rolling horizon MILP solution.

We study the impact of the relative timing of decisions and information flow on the quality of the solutions.

## *4.1 Introduction*

Supply chain management is an important optimization challenge in part because the quality of the current decisions are depended on the future market conditions. Often, information is received during the planning horizon and the likelihood of seeing the future states of the system is altered. This requires that policies for different state trajectories to be found though multi-stage optimization. Formulating and solving such problems generally entails

exploring a large number of scenarios or performing multi-dimensional integrals over probability distributions, which typically increase exponentially with the number of decision stages. A more complete discussion of the complexity of multistage stochastic problems can be found in *Shapiro and Nemirovski* [69].

Multistage planning and scheduling optimization problems have been posed as mathematical programming problems with a single scenario picked for the future, often the expected value of the random parameters [68]. Until recently, given the limitations of the optimization solvers, this was the state of the art.

Recently, mathematical programming and especially MILP formulations with a more refined approach to uncertainty have been a subject of considerable research [19], but the method runs into several bottlenecks:

1. Finding a solution that is a single set of actions misses the opportunity to revise actions depending on the state that the system actually visits. In math programming the goal instead of retrieving a policy that map the states to actions, it is to retrieve numerical actions from the current state.

2. Finding policies requires that the branching of the future scenarios be taken into account, this by itself presents the following problems:

    - The number of scenarios increases very rapidly with the length of the time horizon. Therefore even writing the exact multistage problem as a mathematical program becomes a hard task.
    - One needs fairly restrictive assumptions about how the actions and the future interact. For example, it is very difficult to express situations in which the actions change the nature of the underlying transitions (e.g., by revealing information.).

Our approach to incorporate the need to react to changing information is to adopt a rolling horizon approach Yi and Reklatis [72] [1]. The optimization is resolved at some frequency with the uncertain information updated to reflect new information and also the

[1] This work is is an extension of [73]

actual outcome of the actions implemented. The length of the horizon can be adjusted to balance the computational complexity against the need to preserve an adequate representation of the future.

Rolling horizon MILP takes care the lack of feedback, but runs into a compromise between the first two problems. Essentially, one can limit the combinatorial explosion using the rolling horizon idea, but does not solve the problem of having choices now, that depend on the future and are not properly evaluated.

The above discussion has motivated us to explore Dynamic Programming (DP) as a solution architecture. This requires a different set of compromises compared to mathematical programming approaches. Most specifically, the computational and memory load for exact DP is enormous because the full set of states must be enumerated. Therefore Approximate DP (ADP) has been proposed. What follows are some advantages to an ADP approach. First the underlying representation of the solution is a policy that maps states to actions. This means that once the solution has been computed, the recovery of the actions is simply a lookup table. Second, the underlying computation of state information and transitions is a procedural code, rather than having to be declared in an explicit form in a mathematical program.

Next, we first provide an overview of the problem statement and then the motivation behind our conducted numerical experiments.

### 4.1.1 An Overview Of A Light Aromatic Supply Chain Case Study

In Fig.13, we illustrate a simplified flow diagram of a typical refinery and Fig.15 presents a more detailed diagram of the specific plant configuration.

For the purposes of this work, we will restrict our attention to a typical refining process chain namely the BTX supply chain. Light naptha, a product of the atmospheric distillation process, is the main input for the BTX supply chain. Light naphta is processed in the Reforming unit, where reformate oil is produced. This oil is sent to the BTX extraction unit, where Benzene, Toluene and mixed Xylene is produced. The mixed Xylene is sent to isomerization units (Parex unit) in order for o-Xylene and p-Xylene to be produced.

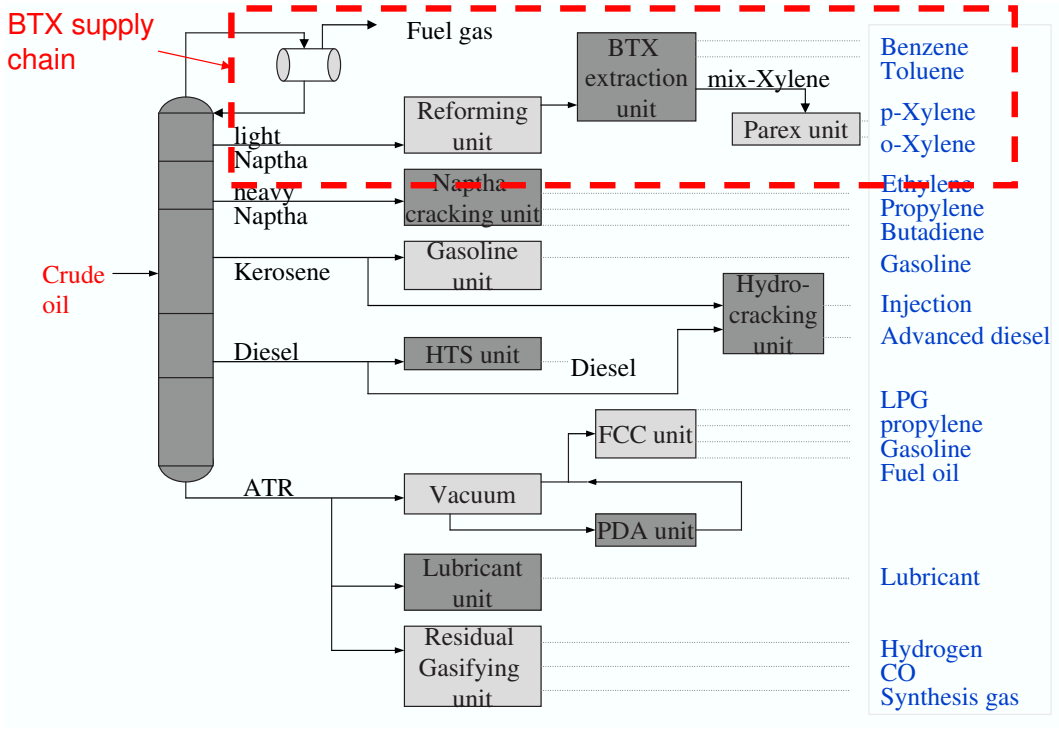



**Figure 13:** The Simplified Flow Diagram of a Typical Refinery.

These five light aromatic products are the main BTX products which will be distributed customers. The uncertainties we choose to focus on are the customer demand and the market price and are modeled via a Markov chain.

### 4.1.2 Motivation Of Our Numerical Studies

The motivation for our numerical experiments is to study the impact of relative timing of decisions and information flow. The BTX supply chain application has two different decisions corresponding to the operational modes of units and the flows to and from the input and output tanks attached to the units. We examine two situations: 1) the mode and flow decisions to occur simultaneously, in other words to share the same time scale and the same information, and 2) the mode decision precedes the flow decisions and is made before certain information is available. Figure 14 demonstrates the relative timing of the decisions, information revelation and state updates in the two cases. The key difference is whether the price and the demand are revealed before or after the mode decisions are to be made.

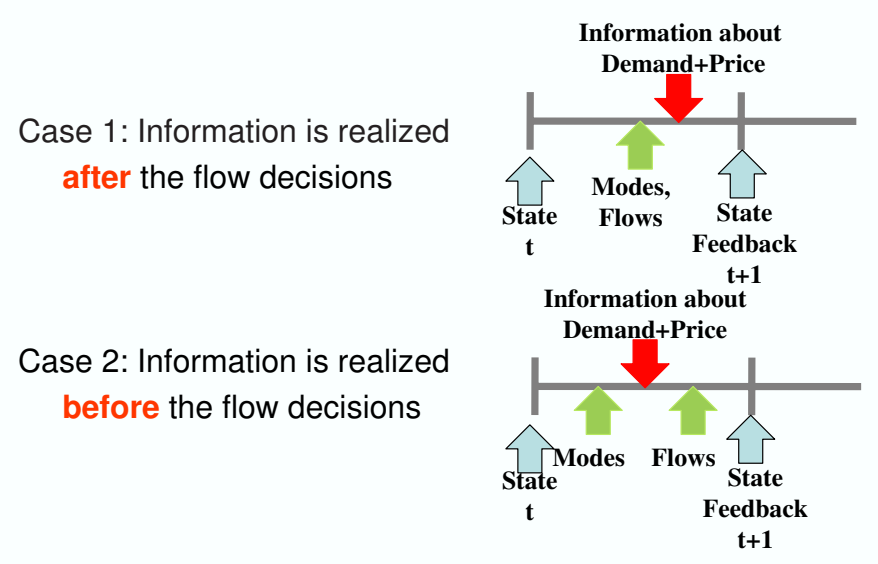



**Figure 14:** Impact of relative timing of decisions and information flow.

The rest of the chapter will present two methodologies for approaching this problem, dynamic programming and rolling horizon MILP. Section 4.2 will present the specific BTX supply chain case study in the form of an MILP and then the uncertainty added to generate a Markov Decision Process (MDP). The case study will be solved by both ADP and rolling horizon MILP and the computational results will be discussed.

## *4.2 Modeling The High Dimensional Supply Chain Case Study As An MILP*

The supply chain case study is inspired from the work of *Kuo and Chang* [74]. The reader can retrieve a detailed MILP model in their paper. We address the simplest possible version of their BTX supply chain (Benzene ($Bz$), Toluene ($Tol$), Mixed Xylenes ($Mx$), P-Xylene ($Px$) and O-Xylene ($Ox$)). Such network is characteristically referred at their work as single train BTX supply chain.

The BTX supply network considered in our case studies is sketched in Fig. 15. The simplifications from the complex case study of *Kuo and Chang* are the following: a) we consider the optimization of a single refinery and not the simultaneous optimization of three refineries, b) the network configuration considered in this work is radically different,

since in *Kuo and Chang* the chain consists out of three separate refineries. Analytically the units involved for the operations of this chain are: three reforming units, three aromatic extraction units, two xylene fractionation units, two (2) tatory units, two parex units and two xylene isomar units, while our structure is composed out of one of each units in order to ensure that such a chain can be a realistic realistic, c) we also assume that there is no capability to export finished product unlike their case study.

### 4.2.1 Introduction

The main units of our BTX supply chain are : 1) One Topping unit denoted as ($Top$), 2) One reformer unit denoted as ($R$) , 3) One BTX or so called extraction unit denoted as ($B$), 4) One Tatory unit denoted as ($TT$), 5) One Xylene Fractorization unit denoted as ($XF$), 6) One Iso Xylene unit denoted as ($I$), 7) One Parex unit denoted as ($P$), 9) One Terminal ($T$) unit, 10) Five customers ($C$) for the main products, and 11) an Import facility unit denoted as ($Im$).

The superstructure of our chain is given and is displayed at Figure 15. For visual purposes the connections between the units and its output tanks are omitted.

The operational role and some further details concerning the chemistry of each unit can be found at [74].

### 4.2.2 Mathematical Modeling of of the Supply Chain

The mathematic description accommodates: a) definition of the sets b) material flow equations c) hard constraints d) objective function.

### 4.2.3 Sets

- $U = \{Im, Top, R, B, TT, XF, I, P, T, C\}$ is defined as the set that contains all the supply chain units. An element of this set is denoted as $u$.

- $F_U = \{F_{Top}, F_R, F_B, F_{TT}, F_{XF}, F_I, F_T, F_C\}$. Each element of $F_U$ is denoted as $F_u$ and is a set that represents the feed tanks of a specific unit $u$. An element of such a set is denoted as $s$. We will define explicitly $s$ for each unit t the next subsection.

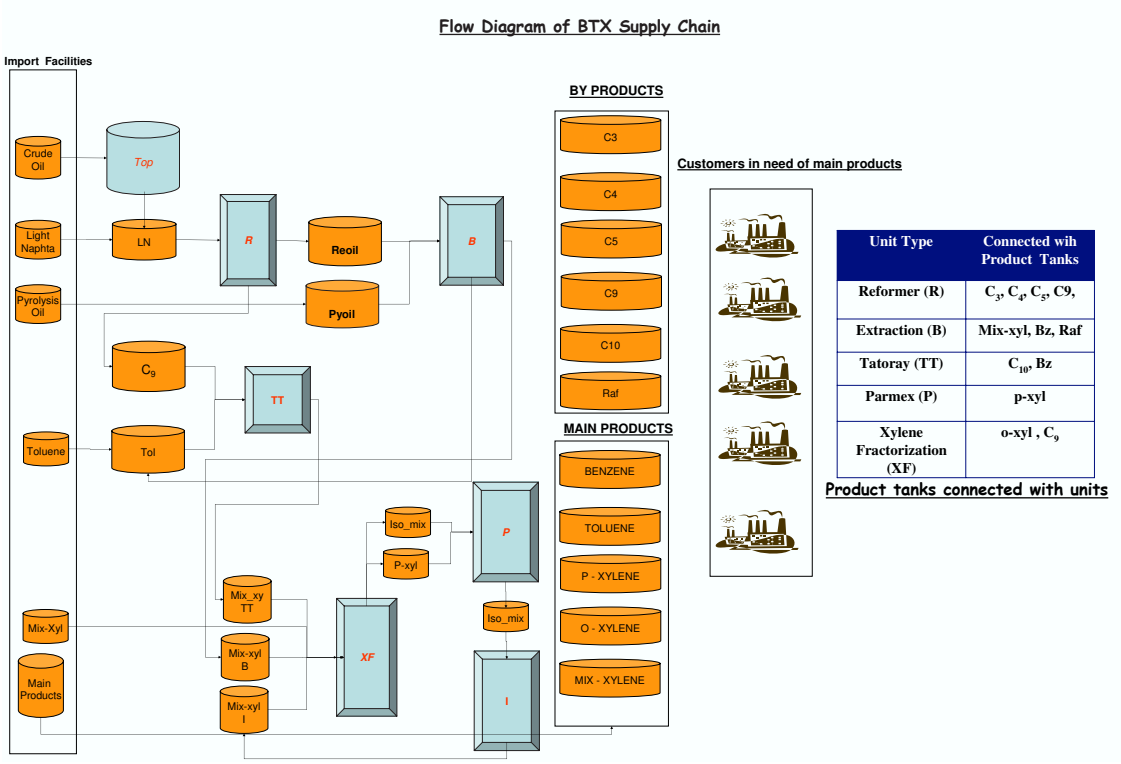



**Figure 15:** Flow Diagram of a simplified BTX Supply Chain .

- $P_U = \{P_{Top}, P_R, P_B, P_{TT}, F_{XF}, F_I, F_T, F_C\}$. Each element of $P_U$ is denoted as $P_u$ is a set that represents the products of a specific $u$. An element of such a set is denoted $p$. We will define explicitly $p$ for each unit at the next subsection.

- $K_R$ is the set that defines the operational tasks that the reformer can perform. $K_{TT}$ is the set that defines the operational tasks that the Tatoray unit can perform.

### 4.2.4 Control Volumes at each Tank

Each unit $u \in U$ has a specific number of tanks that act as its input and a specific number of output tanks. The output tanks are as many as the products of a particular unit. By introducing control volumes at an input and output tanks of each unit we derive from first principles the following equations :

**Control Volume at an input tank of a specific unit** $u$

$$Y_{u,s}(t+1) = Y_{u,s}(t) + \sum_i y_{i,u,s}(t) - x_{u,s}(t) \tag{30}$$
$$\forall i, u \in U, s \in F_u$$

- The inventory at the feed tank $s$ at the next period $(t+1)$ is denoted as $Y_{u,s}(t+1)$.
- The inventories forwarded from the supply chain units $i$ at a particular feed tank $s$ of a unit $u$ at period $(t)$ is denoted as $\sum_i y_{i,u,s}(t)$.
- The amount of inventory that feeds a unit $u$ from a feed tank $s$ at period $(t)$ is denoted as $x_{u,s}(t)$.

If there is no connection $y_{i,u,s}(t)$ from a unit $i$ to the tank $s$ then $y_{i,u,s}(t) = 0$.

**Control Volume at an output tank of a specific unit** $u$

$$Y_{u,p}(t+1) = Y_{u,p}(t) + Pr_{u,p}(t) - \sum_i y_{u,i,p}(t) \tag{31}$$
$$\forall i, u \in U, p \in P_u$$

- The amount of product $p$ stored at the unit $u$ output tank at the next period $(t+1)$ is denoted as $Y_{u,p}(t+1)$.
- The amount of product $p$ produced at the unit $u$, to be stored at the output tank at period $(t)$ is denoted as $Pr_{u,p}(t)$.
- The amount of product $p$ forwarded to the $i^{th}$ unit from the output tank of unit $u$ output tank at period $t$ is denoted as $y_{u,i,p}(t)$.

If there is no connection $y_{u,i,p}(t)$ to a unit $i$ from the tank $p$ then $y_{u,j,p}(t) = 0$.

The active connections regarding each unit are discussed next. We intentionally use a loose repetitive notation concerning each units feed and output tanks. We omit the byproduct tanks, namely $C_3, C_4, C_5, C_9, C_{10}, Raf$ of the terminal unit for space.

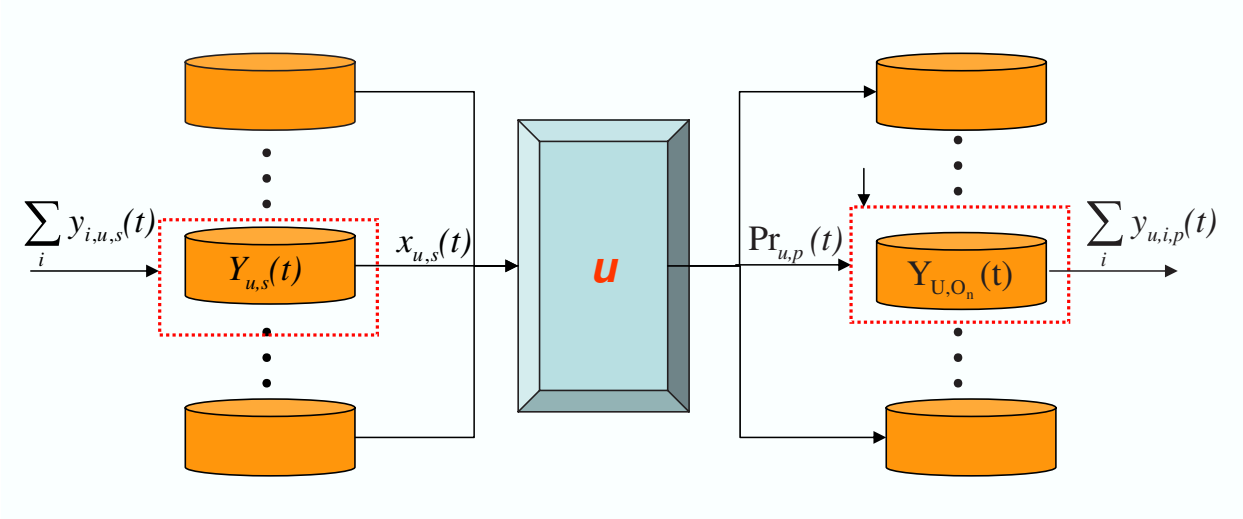



**Figure 16:** Control volumes on the input and output tanks of each unit.

#### *4.2.4.1 Topping unit*

Eq.(32) and Eq.(33) are general equations of the material balance of the input and output tanks of the topping unit.

- Input tank : At this unit $s$ represents the crude oil (CO) tank .

  Therefore: $y_{i,Top,s}(t) = 0 \ \forall i \in U \backslash \{Im\}$.

- Output tank: At this unit $p$ represents the Light naphtha (LN) tank, which coincides with a feed tank of the reformer unit.

#### *4.2.4.2 Reformer unit*

- Input tanks: At this unit $s$ represents the LN tank.

  Therefore: $y_{i,R,s}(t) = 0 \ \forall i \in U \backslash \{Im, Top\}$.

- Output tank: At this unit $p$ coincides with $P_R = \{p_1, p_2, p_3\}$.

  a) $p_1 = \{C_3, C_4, C_5\} \rightarrow y_{R,i,p_1}(t) = 0 \ \forall i \in U \backslash \{T\}$.

  b) $p_2 = \{C_9\} \rightarrow y_{R,i,p_2}(t) = 0 \ \forall i \in U \backslash \{T, TT\}$.

  c) Reformate oil is denoted by $Reoil$. $p_3 = \{Reoil\} \rightarrow y_{R,i,p_2}(t) = 0 \ \forall i \in U \backslash \{B\}$.

  e) $O_5$ contains reformate oil ($Reoil$). For the $O_5$ tank: $y_{R,i,O_5}(t) = 0 \ \forall i \in U \backslash \{B\}$.

d) $O_4$ contains $(C_9)$. For the $O_4$ tank: $y_{R,i,O_4}(t) = 0\ \forall i \in U \backslash \{T, TT\}$.

#### *4.2.4.3 Extraction unit*

- Input tank: We define as $s_i$'s the elements of the $F_B = \{Reoil, Pyoil\}$.

  a) $s_1 = \{Reoil\} \rightarrow y_{i,B,s_1}(t) = 0\ \forall i \in U \backslash \{R\}$.

  b) Pyrolysis oil is denoted by $Pyoil$.

  $s_2 = \{Pyoil\} \rightarrow y_{i,B,s_2}(t) = 0\ \forall i \in U \backslash \{Im\}$. The pyrolysis oil comes from the cracking unit, which is outside the control volume of the studied system. Therefore it is considered as imported stream.

- Input tank: a) $F_1$ contains $Reoil$. For the $F_1$ tank: $y_{i,B,F_1}(t) = 0\ \forall i \in U \backslash \{R\}$.

  b) $F_2$ contains Pyrolysis oil $Pyoil$. For the $F_2$ tank: $y_{i,B,F_1}(t) = 0\ \forall i \in U$.

- Output tank: We define as $p_i$'s the singleton sets that their union forms the $P_B = \{Bz, Tol, Mx, Raf\}$.

  a) $p_1 = \{Bz\} \rightarrow y_{B,i,p_1}(t) = 0\ \forall i \in U \backslash \{T\}$.

  b) $p_2 = \{Tol\} \rightarrow y_{B,i,p_2}(t) = 0\ \forall i \in U \backslash \{TT\}$.

  c) $p_3 = \{Mx\} \rightarrow y_{B,i,p_3}(t) = 0\ \forall i \in U \backslash \{T, XF\}$.

  d) $p_4 = \{Raf\} \rightarrow y_{B,i,p_4}(t) = 0\ \forall i \in U \backslash \{T\}$.

  Output tanks: a) $O_{P_B}$ contains $P_B = \{Bz, Tol, Mx, Raf\}$.

  a) $O_1$ contains $(Bz)$. For the $O_1$ tank: $y_{B,i,O_1}(t) = 0\ \forall i \in U \backslash \{T\}$.

  b) $O_2$ contains $(Tol)$. For the $O_2$ tank: $y_{B,i,O_2}(t) = 0\ \forall i \in U \backslash \{TT\}$.

  c) $O_3$ contains $(Mx)$. For the $O_3$ tank: $y_{B,i,O_3}(t) = 0\ \forall i \in U \backslash \{T, XF\}$.

  d) $O_4$ contains $(Raf)$. For the $O_4$ tank: $y_{B,i,O_4}(t) = 0\ \forall i \in U \backslash \{T\}$.

#### *4.2.4.4 Tatoray unit*

- Input tank: We define as $s_i$'s the elements of the $F_{TT} = \{Tol, C_9\}$.

  a) $s_1 = \{Tol\} \rightarrow y_{i,TT,s_1}(t) = 0\ \forall i \in U \backslash \{B, Im\}$.

  b) $s_2 = \{C_9\} \rightarrow y_{i,TT,s_2}(t) = 0\ \forall i \in U \backslash \{R\}$.

- Output tank: We define as $p_i$'s as singleton set that they union forms the $P_{TT} = \{Bz, Mx, C_{10}\}$.

  a) $p_1 = \{Bz\} \rightarrow y_{TT,i,p_1}(t) = 0 \ \forall i \in U \backslash \{T\}$.

  b) $p_2 = \{Mx\} \rightarrow y_{TT,i,p_2}(t) = 0 \ \forall i \in U \backslash \{XF\}$.

  c) $p_3 = \{C_{10}\} \rightarrow y_{TT,i,p_3}(t) = 0 \ \forall i \in U \backslash \{T\}$.

  Input tank: a) $F_1$ contains $Tol$. For the $F_1$ tank: $y_{i,TT,F_1}(t) = 0 \ \forall i \in U \backslash \{B, Im\}$.

  b) $F_2$ contains $C_9$. For the $F_2$ tank: $y_{i,TT,F_1}(t) = 0 \ \forall i \in U \backslash \{R\}$.

- Output tanks: a) $O_{P_{TT}}$ contains $P_{TT} = \{Bz, Mx, C_{10}\}$.

  a) $O_1$ contains $(Bz)$. For the $O_1$ tank: $y_{TT,i,O_1}(t) = 0 \ \forall i \in U \backslash \{T\}$.

  b) $O_2$ contains $(Mx)$. For the $O_2$ tank: $y_{TT,i,O_2}(t) = 0 \ \forall i \in U \backslash \{XF\}$.

  c) $O_3$ contains $(C_{10})$. For the $O_3$ tank: $y_{TT,i,O_3}(t) = 0 \ \forall i \in U \backslash \{T\}$.

#### *4.2.4.5 Xylene Fractorization unit*

- Input tank: We define as $s_i$'s the singleton of the $F_{XF} = \{Mx\}$.

  a) $s_1 = \{Mx\} \rightarrow y_{i,XF,s_1}(t) = 0 \ \forall i \in U \backslash \{B, Im, I, TT\}$.

- Output tank: We define as $p_i$'s the singleton sets that they union forms the $P_{XF} = \{Px, Ox, C_9, Ix\}$.

  a) $p_1 = \{Px\} \rightarrow y_{XF,i,p_1}(t) = 0 \ \forall i \in U \backslash \{P\}$.

  b) $p_2 = \{C_9\} \rightarrow y_{XF,i,p_2}(t) = 0 \ \forall i \in U \backslash \{T\}$.

  c) $p_3 = \{O_x\} \rightarrow y_{XF,i,p_3}(t) = 0 \ \forall i \in U \backslash \{T\}$.

  d) $p_4 = \{I_x\} \rightarrow y_{XF,i,p_4}(t) = 0 \ \forall i \in U \backslash \{P\}$.

- Input tank: a) $F_1$ contains $Mx$. For the $F_1$ tank: $y_{i,XF,F_1}(t) = 0 \, \forall i \in U \backslash \{TT, I, B, Im\}$.

- Output tanks: a) $O_{P_{XF}}$ contains $P_{XF} = \{Px, Ox, C_9, Ix\}$.

  a) $O_1$ contains P-xylene $(Px)$. For the $O_1$ tank: $y_{XF,i,O_1}(t) = 0 \ \forall i \in U \backslash \{P\}$.

  b) $O_2$ contains $(C_9)$. For the $O_2$ tank: $y_{XF,i,O_2}(t) = 0 \ \forall i \in U \backslash \{T\}$.

  c) $O_3$ contains O-xylene $(Ox)$. For the $O_3$ tank: $y_{XF,i,O_1}(t) = 0 \ \forall i \in U \backslash \{T\}$.

d) $O_4$ contains Isomarf ($Ix$). For the $O_4$ tank: $y_{XF,i,O_1}(t) = 0\ \forall i \in U \backslash \{P\}$.

#### *4.2.4.6 Parex unit*

- Input tank: We define as $s_i$'s the singleton sets of the $F_P = \{Ix, Px\}$.

  a) $s_1 = \{Px\} \rightarrow y_{i,P,s_1}(t) = 0\ \forall i \in U \backslash \{XF\}$.

  b) $s_2 = \{Ix\} \rightarrow y_{i,P,s_2}(t) = 0\ \forall i \in U \backslash \{XF\}$.

- Output tank: We define as $p_i$'s the singleton sets that they union forms the $P_P = \{Px, Ix\}$.

  a) $p_1 = \{Px\} \rightarrow y_{P,i,p_1}(t) = 0\ \forall i \in U \backslash \{P\}$.

  b) $p_2 = \{Ix\} \rightarrow y_{P,i,p_2}(t) = 0\ \forall i \in U \backslash \{I\}$.

#### *4.2.4.7 Isomar unit*

- Input tank: We define as $s_1 = Ix$'s since $F_I = \{Ix\}$.

  a) Therefore $y_{i,I,s_1}(t) = 0\ \forall i \in U \backslash \{P\}$.

- Output tank: $p_1 = \{Mx\}$'s since $P_P = \{Mx\}$ is singleton. Therefore $y_{I,i,p_1}(t) = 0\ \forall i \in U \backslash \{XF\}$.

- Input tank: a) $F_1$ contains $Ix$. For the $F_1$ tank: $y_{i,I,F_1}(t) = 0\ \forall i \in U \backslash \{P\}$.

- Output tanks: a) a) $O_1$ contains ($Mx$). For the $O_1$ tank: $y_{I,i,O_1}(t) = 0\ \forall i \in U \backslash \{XF\}$.

#### *4.2.4.8 Terminal*

- Input tank: We define as $s_i$'s the singleton sets of the $F_P = \{Ix, Px\}$.

  a) $s_1 = \{Px\} \rightarrow y_{i,P,s_1}(t) = 0\ \forall i \in U \backslash \{XF\}$.

  b) $s_2 = \{Ix\} \rightarrow y_{i,P,s_2}(t) = 0\ \forall i \in U \backslash \{XF\}$.

- Output tank: We define as $p_i$'s the singleton sets that they union forms the $P_P = \{Px, Ix\}$.

  a) $p_1 = \{Px\} \rightarrow y_{P,i,p_1}(t) = 0\ \forall i \in U \backslash \{P\}$.

  b) $p_2 = \{Ix\} \rightarrow y_{P,i,p_2}(t) = 0\ \forall i \in U \backslash \{I\}$.

- Input tanks: The set of byproducts is denoted by $P_b = \{C_3, C_4, C_5, C_9, C_{10}, Raf\}$ and the set of main products

  by $P_m = \{Bz, Tol, Mx, Ox, Px\}$.

  Byproduct tanks: a) $s_1 = \{C_3, C_4, C_5\} \rightarrow y_{i,T,s_1}(t) = 0 \ \forall i \in U \backslash \{R\}$.

  b) $s_2 = \{C_9\} \rightarrow y_{i,T,s_2}(t) = 0 \ \forall i \in U \backslash \{R, XF\}$.

  c) $s_3 = \{C_{10}\} \rightarrow y_{i,T,s_3}(t) = 0 \ \forall i \in U \backslash \{TT\}$.

  d) $s_4 = \{Raf\} \rightarrow y_{i,T,s_4}(t) = 0 \ \forall i \in U \backslash \{B\}$.

  Main product tanks:

  e) $s_5 = \{Bz\} \rightarrow y_{i,T,s_5}(t) = 0 \ \forall i \in U \backslash \{B, TT, Im\}$.

  f) $s_6 = \{Tol\} \rightarrow y_{i,T,s_6}(t) = 0 \ \forall i \in U \backslash \{B, Im\}$.

  h) $s_7 = \{Ox\} \rightarrow y_{i,T,s_7}(t) = 0 \ \forall i \in U \backslash \{XF, Im\}$.

  i) $s_8 = \{Px\} \rightarrow y_{i,T,s_8}(t) = 0 \ \forall i \in U \backslash \{P, Im\}$.

  k) $s_9 = \{Mx\} \rightarrow y_{i,T,s_9}(t) = 0 \ \forall i \in U \backslash \{TT, Im\}$.

- Outputs : The variables $y_{T,C,(Pm \cup Pb)}(t)$ are decision variables.

  a) $y_{T,C,Pm}(t) = D_{Pm}(t)$, where $D_{Pm}(t)$ is the demand of the main products at time $(t)$.

  b) $y_{T,C,Pb}(t) = D_{Pb}(t)$, where $D_{Pb}(t)$ is the demand of the

#### *4.2.4.9 Constraints on $y_{T,C,(Pm)}(t)$*

We follow the mathematical formulation of *Kuo and Chang* [74] and allow the main product amounts shipped to the customers to deviate from the requested demand values $D_{c,P_m}$. To represent the amounts of backlog and surplus, we introduce two corresponding variables $L_{C,Pm}(t)$ and $E_{C,Pm}(t)$ into the following material balance equation:

$$D_{c,p}(t) - L_{c,p}(t) + E_{c,p}(t) = y_{T,c,p}(t) \tag{32}$$

$$\forall c \in C, p \in P_m$$

At the end of the horizon the total product demand for each customer should be satisfied. This is ensured by:

$$\sum_{t=0}^{h} D_{c,p}(t) = \sum_{t=0}^{h} y_{T,c,p}(t) \tag{33}$$
$$\forall c \in C, p \in P_m$$

The parameters that prescribe the acceptable amount of surplus or backlog are the quantities $a, b$:

$$L_{c,p}(t) \leq aD_{c,p}(t)NB_{c,p}(t) \tag{34}$$
$$\forall c \in C, p \in P_m$$

$$E_{c,p}(t) \leq bD_{c,p}(t)NE_{c,p}(t) \tag{35}$$
$$\forall c \in C, p \in P_m$$

, where $NE_{c,p}(t)$ and $NB_{c,p}(t)$ are binary decision variables.

### 4.2.5 Reaction and Separation Processes - The Determination of $Pr_{u,p}(t)$

The supply chain units are either reaction or separation units. We quantify the product stream $Pr_{u,p}(t)$ for the output tanks $p$ of each unit via the following equations.

#### *4.2.5.1 Reaction Processes*

The set $Ua = \{Top, TT, I, R\}$ represents all the reaction processing units within the system. It should be noted that the reaction yield of every product of each unit is assumed to be dependent upon the chosen operational mode and the total flow proportional to the feed quantity. The generalized material balance of the reaction processes can be written in bilinear form as:

$$Pr_{u,p}(t) = \sum_{s \in F_s} \sum_{k \in K_{u,s}} x_{u,s}(t) Bin_{u,k}(t) YD_{u,s,k,p(t)} \tag{36}$$
$$\forall u \in Ua, \forall u \in P_u, \forall k \in K_{u,s}$$

, where $F_s$ is the set of all allowable feeds of unit $u \in U_a$.

The $K_{u,s}$ set represents the operational modes $k$ of unit $u$ for processing feedstock $s$. $Bin$ are binary variables. There variable enforce that each feed $s$ will be processed by a

certain operational mode. $YD_{u,s,k}(t)$ is a system parameter describing the reaction yield of product $p$, given the input tank $F_s$ with operational mode $k$. This bilinear constraint can be converted to a linear as shown in [74]. Simply, one has to introduce new continuous variables that would physically represent this bilinear form along with logical big- M constraints.

#### *4.2.5.2 Separation Processes*

The set of $U_b = \{B, XF, P\}$ defines all the units within the chain that perform separation processes.

The linear equation that quantifies $Pr_{u,p}(t)$ for these units is:

$$Pr_{u,p}(t) = \sum_{s \in F_s} x_{u,s}(t) RF_{u,s}(t) \tag{37}$$
$$\forall u \in Ub, \forall p \in P_{Ub}$$

, where $RF_{u,s}(t)$ represents design parameters for each of these units.

### 4.2.6 Constraints

*Kuo and Chang* [74] define the operability of the supply chain is ensured, if at least one the reforming and one isomar units are in operation at each time period. For our version of this example we will ensure with logical constraints that every unit must be in operation at each time period, and only one operation mode can be adopted for the feed streams in each unit. In this version of the problem, we assume five operational modes for the reformer unit and three modes for the Tatoray unit.

The constraints associated with the MILP formulation are with respect to: a) Input flow constraints of the units , b) Transportation constraints and c) Constraints on imported materials.

**Input Flow constraints**

$$FB^L_{u,m} \leq \sum_s x_{u,s}(t) \leq FB^U_{u,m} \tag{38}$$
$$\forall u \in Ub, \forall s \in F_u$$

, where $FB^L_{u,m}, FB^U_{u,m}$ are system parameters, low and upper bounds that represent low and upper bounds on the input flows of the separation units.

The hard input flow constraint for the reaction processes are:

$$FB^L_{u,m} \leq \sum_{s \in F_s} \sum_{k \in K_{u,s}} x_{u,s}(t) S_{u,i,k}(t) \leq FB^U_{u,m} \tag{39}$$

$$\forall u \in Ua, \forall s \in F_s$$

**Transportation constraints** The process materials are transferred from a unit to another. The corresponding transportation capacities are in practice, the following inequalities are included in the model:

$$y^L_{u,i,p} \leq y_{u,i,p}(t) \leq y^U_{u,i,p} \tag{40}$$

$$\forall u, i \in U, \forall p \in P_u$$

, where $y^L_{u,i,p}$,$y^U_{u,i,p}$ are design parameters.

**Constraints on the Imported materials**

The model also includes constraints on the quantity of the imported materials. Those are expressed via the following inequalities:

$$y^L_{Im,u,s} \leq y_{Im,u,s}(t) \leq y^U_{Im,u,s} \tag{41}$$

$$\forall u \in U, \forall s \in F_u$$

, where $y^L_{Im,i,s}$,$y^U_{Im,i,s}$ are system parameters.

### 4.2.7 Decision Variables

The decision variables for this problem are:

- $y_{i,u,s}(t) \forall i, u \in U, \forall s \in F_u$ : represent the inventory that we choose to forward from a unit $i$ to the particular feed tank $s$ of unit $u$.
- $y_{u,i,p}(t) \forall i, u \in U, \forall p \in P_u$ : represent the inventory that we choose to forward from the output tank $p$ of a particular unit $u$ to other units $i$.
- $x_{u,s}(t) \forall u \in U, \forall s \in F_u$ : represent the inventory that we choose to feed unit $u$ from feed tank $s$.

- $Bin_{u,k}(t) \forall u \in \{R, TT\}, \forall k \in K_{u,s}$ : represents the selected modes at the Reformer or the Tatorray at time t

- $NE_{c,p}(t)$ and $NB_{c,p}(t), \forall c \in C, \forall p \in P_u$ : represents whether we should provide surplus or backlog to a given customer at a given time period.

### 4.2.8 Objective Function

The MILP takes into consideration the dynamics of the system under a specific scenario and maximizes the reward over a specified horizon $h$.

$$max\{\sum_{t=1}^{h} r(t)\} \tag{42}$$

,where $r(t)$ is the myopic reward. Specifically $r$ is the net profit:

$$r(t) = p(t) - c(t), \forall t \in h \tag{43}$$

#### *4.2.8.1 Net Profit $p(t)$*

This variable $r(t)$ represents the net income for this system at each time period. The revenue produced at time $t$ is represented by:

$$p(t) = \sum_{c} \sum_{p} y_{T,c,p}(t) SP_{T,c,p}(t) \tag{44}$$
$$\forall p \in (Pm \cup Pb), c \in C$$

, where $SP(t)$ is the selling price of each product at time $t$.

#### *4.2.8.2 Net Cost $c(t)$*

There are five sources of cost terms for this system. These are: a) the total cost of imported raw materials ($Cr(t)$), b) the total operation cost ($Co(t)$) of each unit, c) the total transportation cost ($Ct(t)$), d) the total inventory cost ($Cs(t)$), e) the total backlog cost ($Cb(t)$). These are described by the following equations Eq.(48-52).

**Cost of Imported Materials**

$$Cr(t) = y_{Im,R,s}(t) CLN(t) + y_{Im,B,s}(t) CPyoil(t) + y_{Im,TT,s}(t) CTol(t) \tag{45}$$
$$+ y_{Im,XF,s}(t) CMx(t) + \sum_{p} y_{Im,T,s}(t) CP_m(t)$$

, where $CLN(t), CPyoil(t), CTol(t), CMx(t), CPm(t)$ is the cost of importing respectively light naphtha, pyroysis gas, toluene, Mixed xylene, main products.

**Operational Cost of each unit**

$$Co(t) = \sum_{u} \sum_{s} x_{u,s}(t) Cf_{u,s}(t) \tag{46}$$
$$\forall s \in F_s, \forall u \in (Ua \cup Ub)$$

, where $Cf_{u,s}(t)$ is the operational cost of each input stream $s$ of a unit $u$.

**Storage Costs**

$$Cs(t) = \sum_{u} \sum_{s} Y_{u,s}(t) Cs_u(t) + \sum_{u} \sum_{p} Y_{u,p}(t) Cp_u(t) \tag{47}$$
$$\forall s \in F_u, \forall p \in P$$

, where $Cs_u(t)(Cp_u(t))$ is the storage cost of a particular feed (product) $s(p)$ at unit $u$.

**Cost of Transported Materials between units**

$$Ct(t) = \sum_{u} \sum_{i} \sum_{p} y_{u,i,p}(t) Ct_{u,i}(t) \tag{48}$$
$$\forall u, i \in U, \forall p \in P$$

, where $Ct_{u,i}(t)$ the transportation cost moving material $p$ from unit $u$ to $i$.

**Backlog Cost**

$$Cb(t) = \sum_{u} \sum_{p} (D_{c,p}(t) - y_{T,c,p}) Cbm_p(t) \tag{49}$$
$$\forall c \in C \forall p \in P_m$$

, where $Cbm_p(t)$ is the backlog cost of each main product.

### 4.2.9 A 2 Stage Stochastic Programming Formulation

In the second numerical case study, it is assumed that the decision space is composed out of decisions with two different time scales. The formulation follows the classic 2 stage stochastic programming as shown in the first chapter of *Birge and Louveaux*[68] with finite scenario support. The first stage decision is the operating modes, while the second decision are the flows that are tailored to the realized scenario.

## *4.3 Formulating the Problem as an MDP*

The formulation of this problem as an MDP requires specification the following elements: State variables, exogenous information variables, decision variables, transition function, one stage profit function, and objective function. The following subsections detail each of these.

### 4.3.1 State Variables / Exogenous Information Variables

For the BTX supply chain instance modeled as an MDP, the state variable is a high dimensional vector defined below:

$$s_t = \begin{bmatrix} Y_{u,s}(t) \\ Y_{u,p}(t) \\ \hat{SP}_{c,p}(t) = \text{ Realized selling price of the main products to customers at time } t. \\ \hat{D}_{c,p}(t) = \text{ Realized demand rate of customers at time } t. \end{bmatrix} \tag{50}$$

In this study the random variables are modeled with a first order Markov model. The models that govern the probabilistic transitions among the given discrete sets of values are completely specified by the probability matrices $P_{D_p,SP_p}$ for $D$ and $SP$. For example, the $(i,j)^{\text{th}}$ element of $P_{D_p,SP_p}$ is the probability of the demand and price taking the next value of $d_{p,j}$ and $sp_{p,j}$ at the next time period from the current value of $d_{p,i}$ and $sp_{p,i}$. By $d_{p,i}$ we denote the $i^{th}$ demand realization of a product $p$.

Realized values of the random variables at time $t$, $\hat{D}_{c,p}(t)$ and $\hat{SP}_{T,c,p}(t)$, are assumed to be known to the decision-maker. Hence they constitute exogenous information variables. However, the values of these variables at future times are uncertain and are described by the corresponding probability distributions. It is customary to include the parameters defining these conditional probability distributions of the random variables as a part of the state vector as they capture the information available at that time period. They are therefore called information states. With a first order Markov chain model, it is sufficient to include just $\hat{SP}_{c,p}(t)$ and $\hat{D}_{c,p}(t)$ in the system state as they completely define the conditional probability distributions of the future random variables.

### 4.3.2 Decision Variables

The decision variables of the MDP coincide with the decision variables of the MILP formulation.

### 4.3.3 Transition Function

The key element in modeling the systems state transition is to take into account the physical constraints as described in Eq.(33) and Eq.(34) among the interdependent units and also the probabilistic state transitions of the random variables.

The stochastic counterpart of the state definition is the $SP_{T,c,p}$ and $D_{c,p}$. These random variables are modeled with a first order Markov model. This probabilistic model governs the transitions of the probability distributions among given discrete sets of values. In our numerical illustration this model has 4 states for all the main products except the Mixed Xylene product, which is described with 2 states.

The chosen states are : 1) Low demand - Low price, 2) High Demand - Low price, 3) High Demand - High Price, 4) Low Demand - High Price.

The reasoning behind this state representation is that quantitatively given low demand for a product and low price, we anticipate with high probability the transition to a high demand associated with the low price. Similarly, the high demand could stimulate an increase in price, followed by a fall in demand and the price. This cycle is proposed as illustration of coupled demand and price dynamics. Such patterns can be represented using first order Markov models based on a wide variety of environmental variables.

#### *4.3.3.1 Contribution (Profit) Function*

The one step profit produced by a decision $\alpha_t$ at state $s_t$ during one time period with random variable $\omega_{t+1}$ is denoted as $\hat{f}(s_t, \alpha_t, \omega_{t+1})$ which coincides the one stage profit of the MILP formulation. With some abuse of notation we denote $\omega_{t+1} = [\hat{D}_{c,p}(t), \hat{SP}_{c,p}(t)]$, which is a five dimensional vector describing the outcome of the independent Markovian processes at time $t+1$. Then, the expression for $f(s_t, \alpha_t)$ is:

$$f(s_t, a_t) = \mathbb{E}[\hat{f}(s_t, a_t, \omega_{t+1})] = \sum_{j=1}^{N} P(s_j|s_t, \alpha_t)\hat{f}_j(s_t, \alpha_t, \omega_j) \tag{51}$$

### 4.3.4 Objective Function

The objective in this problem is to maximize the discounted expected profit over an infinite horizon. This will be accomplished, when we find a stationary[2] decision function $\pi(s_t)$ such that each state is mapped to the best possible action. The total return of a policy $\pi$ starting from an initial state $s_0$ is:

$$F_{s_0}^{\pi} = \sum_{t=0}^{\infty} \gamma^t f(s_t, \pi(s_t))|s_0) \tag{52}$$

$f(s_t, a_t)$ is defined at section 3.7. Trivially, the optimal policy $\pi^*$ as discussed before is the one instructed by the optimal value function:

$$\pi^* = \arg\{max_{\pi \in \Pi} F_{s_0}^{\pi}\} \tag{53}$$

## *4.4 Information Flow And Decision Making*

The following architectures are designed to study the impact of the relative timing of decisions and information flow.

For the case where the mode and flow decisions occur simultaneously we will use: a) Rolling Horizon MILP With Sampling From The Probability Distributions , b) Rolling Horizon MILP With The Most Probable Scenario, c) a variance of a real time dynamic programming approach named RTADP delineated at Chapter 3.

For the case, where the mode decision precede the flow decisions we will use: a) A Rolling Horizon 2 Stage Stochastic Programming Approach, b) the RTADP approach.

### 4.4.1 A Real Time Approximate Dynamic Programming Algorithm

Its intuitive that for strategic planning applications we cannot solve for the optimal value functions for the entire state space because of the curse of dimensionality. First, *Barto et al* [15] introduces the important concept of the relevant state space. Assuming that we knew the optimal policy, if it was to be simulated, the encountered states would consist the

[2]There is ambiguity regarding the nature of the resulting policy (stationary or not stationary). Since, if one uses ADP methods that utilize only a vanishing fraction of the entire state space the resulting policy may not be stationary [75].

relevant state space envelope. In other words a state is called relevant if it can appear with positive probability during the application of an optimal policy.

If we were to perform approximate value iteration as explained by *Lee et al* [64] to this envelope of states we would retrieve the optimal policy. Essentially this represents the fact that we do not have to consider the entire $\mathbb{S}$ to come up with a close to the optimal policy, but we need a methodology that samples selectively the state space.

Studying some proposed ADP architectures [17, 18, 64], we notice that they solve for the value functions only for the regions that the system normally operates. To identify such regions of the state space they would implement from different initial states heuristic rules via simulation of the stochastic system and they store in a list the system states they encountered. This list of states is often called value table

Our strategy is to propose modifications to the existing RTDP approach, which will construct the value table from scratch. In principle this approach will learn a high quality policy and continuous improve on it via episodic learning. Episodic learning is usually defined as a learning procedure where individuals strategies are updated after each episode. Each episode is a simulation of a particular policy for a determined horizon. At each episode sufficient statistics regarding the different action payoffs are accumulated. In this particular case we consider a single agent (us) and the quality of the statistics to be the value function estimation for the sampled states.

Out of the scope of this article, but an intresting fact is that if the problem involves other agents as well, the optimal policy of an agent would be a Markovian randomized policy, where the individual agent policy wouldbe a probability distribution over a set of actions. In principle, the RTADP can be applied to a multi agent setting, if we include as the environment apart from the exogenous uncertainty (random variables) all of the other agents. It fact a similar architecture has been applied recently with success in tracking moving-evasive targets [76]. Recent advances in Markov Models for Multi-Agent Coordination can be retrieved in [77] [78] [79].

A high level description of the RTADP procedure follows. First, we start given an empty state space or value table and we gradually fill it with entries using an $\epsilon$-greedy

episodic RTDP. From each state,we do not evaluate the entire action space, but only a set of actions named adaptive action set. This concept has been seen in the previous chapter and its purpose is to alleviate the COD with reprect to the action space and also to utilize deterministic mathematical programming as a mean to provide close to optimal actions to a stochastic setting. Our approach uses the initialization of the value function for the unseen successive states as a parameter to tune the exploration vs exploitation trade off. By the term exploration, we mean to explore the state space, having as goal to retrieve more states and improve on the performance of our policy. The term exploitation, means to use greedy actions based on the existing value table knowledge to maximum gain value. Last, we suggest the usage of a local neighbor approximator that if used within an Approximate Value Iteration architecture [64], it properly defines a contraction mapping.

RTDP works in episodes, meaning that each episode starts from a starting state within the value table and accumulates valuable learning along its trajectories. It is empirically proven that after a finite number of trials the visited states that are involved in trial trajectories become saturated and belong to a closed set of states. We refer the reader to [63] for the definition of the simulated relevant state space denoted as $S_{\text{sim}}$: This set contains all the states that belong to the trajectory of the optimal policy. From a practical standpoint, there is no algorithm that can exactly identify this set of states for general problems.

### 4.4.2 The RTADP Algorithm

The procedure below samples the state space using a greedy policy and constructs a value table denoted as $\mathbb{S}_{\text{sim}}$ starting from an empty one by gradually adding entries, as states are encountered in the simulation. The following steps are involved in each iteration of the algorithm and a schematic representation appears in Figure 17.

**For episodes** $j = 1, 2, ...M$ , where $M$ is a sufficiently large integer

**For iterations** $i = 1, 2, ...h$ , where $h$ is the horizon length of each episode

**Step 1** Start from a *random* state $s_t \in \mathbb{S}_{sim}$.

**Step 2** Construct set of actions (denoted by $A_{sub}$) for $s_t$. $A_{sub} \subset A$, where $A$ is the set of

all possible controls that the decision maker can exercise at any time instance for a given state.

**Step 3** Update the value function of $J(s_t)$:

$$J(s_t) = \max_{\alpha \in \mathbb{A}_{sub}} \{f(s_{t+1}, \alpha_t) + \gamma \sum_{s_{t+1} \in \mathbb{S}} P(s_{t+1}|s_t, \alpha_t) J(s_{t+1}) | S_t = s_t\} \tag{54}$$

$$\alpha^* = \mathbf{arg} \max_{\alpha \in \mathbb{A}_{\sim \approxeq}} \{f(s_{t+1}, \alpha_t) + \gamma \sum_{s_{t+1} \in \mathbb{S}} P(s_{t+1}|s_t, \alpha_t) J^i(s_{t+1}) | S_t = s_t\} \tag{55}$$

**Step 4** A state $s_{t+1}$ is sampled according to the probability distribution $P(s_{t+1}|s_t, \alpha^*(s_t))$ as defined from the Markov model of the random variables. Set $t = t + 1$ and we go back to Step 1.

**End**

**End**

Note that, if the algorithm happens to circulates over a small cyclic graph of states, the algorithm is restarted from a *random* state $s_i \in \mathbb{S}_{sim}$. Empirically, one terminates with : $(\|J^{\pi}_{i+1}(s_i) - J^{\pi}_i(s_i) < \varepsilon\|_{\infty} \quad \forall s_i \in \mathbb{S}_{sim} \subseteq \mathbb{S}$ ) like in VI, where $\varepsilon$ is a tolerance parameter. The user can apply this termination criterion, only if the state space is saturated and the number of entries does not grow.

One can try different updating value function schemes, while implementing this RTADP algorithm. For instance assume that the set of states to be updated in episode $j$, namely $X_j$, is generated by simulation. Because MDPs are acyclic, we apply prioritized sweeping, which means that after each iteration, the profit-to-go estimations are updated in the reverse order in which they appeared during the simulation. Assume, for example, that $X_j = \{x^j_1, x^j_2, ..., x^j_h\}$. In this case the order in which the updates are performed, is $x^j_h, ..., x^j_1$.

### 4.4.3 Key Elements of $A_{sub}$

$\mathbb{A}_{sub}$ is composed out of the following sources:

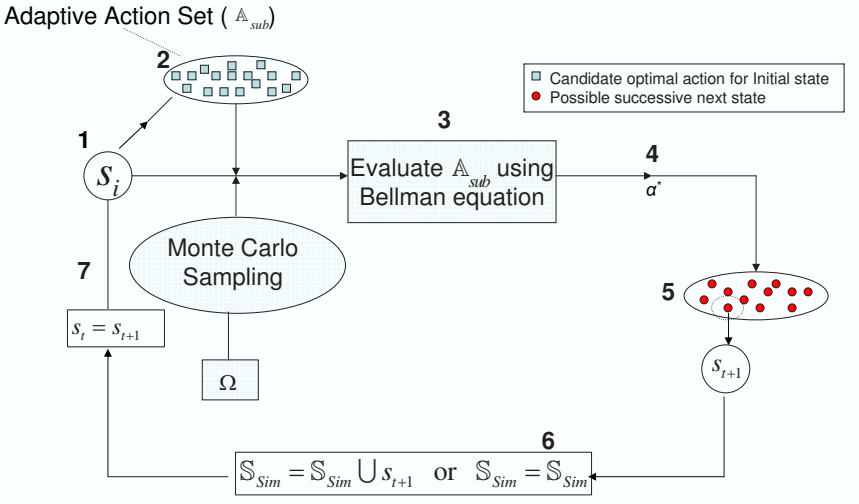



**Figure 17:** Schematic representation of sequential calls on RTADP algorithm .

1. *Mathematical Programming actions:* If we can describe the problem of interest with a deterministic mathematical model(e.g., MIP), then it is highly advisable to try actions resulting from *deterministic* math program formulations. This typically constitutes a suboptimal policy.

2. *Best known actions:* This action is a product of the a-priori learning with respect to the value functions of all states in the "evolving" value table. If the state to be updated is a state never visited before, then its best known action is empty. If the state has been revisited, a best known action should have been stored with respect to the prior estimate of the value function.

3. *Random actions:* Random controls ensure that we effectively explore the entire action space and exclude the possibility of not visiting any portion of the state space. We generated the random controls by random perturbations to the heuristic and mathematical programming actions for state $s_i$.

4. Other candidate optimal actions, as a part of the $A_{sub}$, are the best known actions of the $k - NN$ of state $s_i$ o.

### 4.4.4 Calculating $J(s_t)$

Exact DP techniques (e.g.,VI, PI) can be applied only to MDP's with finite $\mathbb{S}$, $\mathbb{A}$ and require an initial estimation of the value function $\forall s \in \mathbb{S}$. The optimal policy created by these techniques is not sensitive to any initial estimation scheme. Since the proposed approach does not consider the entire state space in order to address large MDP problems, it foregoes the formal convergence of the value functions. This section provides rules of thumb for structuring such an estimation scheme, so as to achieve empirical convergence and maximize the performance.

In Eq.(54) the calculation of $J_t^\pi(s_t)$ involves the knowledge of the value function of all possible successive states $J_t^\pi(s_{t+1})$ for each action in $A_{sub}$. During this calculation we will encounter one of these three possible scenarios. A schematic representation of the following scenarios appears in Figure 18.

- Scenario 1: All $\{s_{t+1}\}$ have values registered in the value table. We use these values to calculate $J_t^\pi(s_t)$.

- Scenario 2: Some of the $\{s_{t+1}\}$s are not found in the value table.In this case we first need to find the set of states within $\delta$ distance of $s_{t+1}$ (denoted by $\mathcal{N}_\delta(s_{t+1})$). Here we use the Euclidean distance metric $d$, as proposed by Lee and Lee [33], with a user defined design parameter $\delta$.

$$Find \quad \mathcal{N}_\delta(s_{t+1}) \stackrel{\text{def}}{=} \left\{ s \in S \ : \ d = \sqrt{(s - s_{t+1})^T W (s - s_{t+1})} < \delta \right\} \tag{56}$$

  , $W$ is a feature weighting diagonal matrix. If $|\mathcal{N}_\delta(s_{t+1})| \geq k$, then we can approximate the value function of $s_{t+1}$ from the $k$ nearest states, by utilizing a local $k$-nearest neighbor approximator. The mathematical expression for approximating the value function for each $s_{t+1}$ is as follows:

$$J_t^\pi(s_{t+1}) = \frac{1}{k} \sum_{x \in \mathcal{N}_k(s_{t+1})} J_t^\pi(x) \tag{57}$$

  where $\mathcal{N}_k(s_{t+1})$ denotes the set of states representing the k nearest neighbors. The value $k$=4 was used for the numerical results, where the authors proved the numerical

stability and convergence of the VI in various of applications using the $k - NN$ as approximation scheme.

In the case that $s_j$ has $k' < k = 4$ neighbor states within $\mathbb{S}_{REL}$, we utilize Eq.(60) with the $k'$ number states to approximate $J_i^{\pi}(s_j)$.

- Scenario 3: The case where $|\mathcal{N}_{\delta}(s_{t+1})| = 0$. Apparently Eq.(60) cannot be used, and therefore we suggest an initial estimation scheme of underestimating the value functions with respect to the optimal one. Nonetheless, in this application the $J_t^{\pi}(s_{t+1})$ of the $s_{t+1}$s that belong to scenario 3 are initialized with 0. In the previous chapter we studied the exploration vs exploitation due to the initialization with respect to the third scenario. Taking this further, the decision-maker may introduce a priori knowledge, similar to *Choi et al*[17, 18].

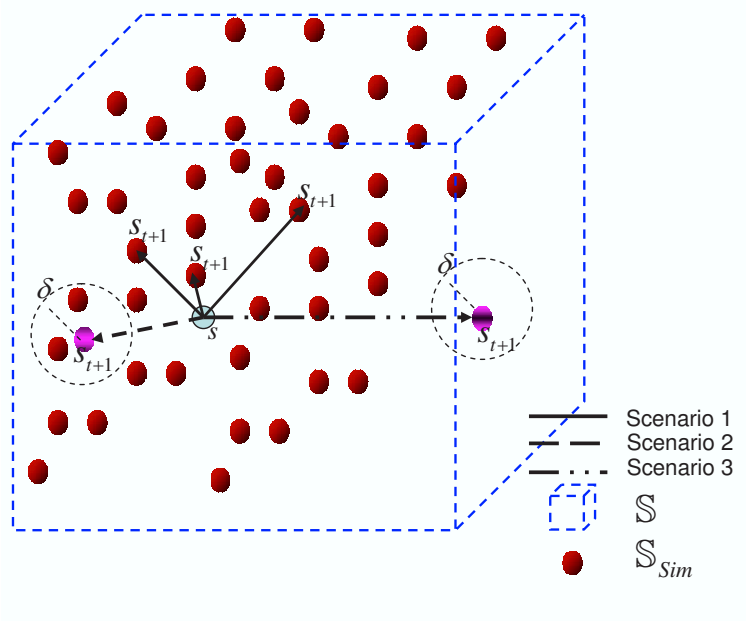



**Figure 18:** Schematic representation of the 3 scenarios corresponding to a legal state transition inside the state space. $\mathbb{S}$ is the entire state space. $\mathbb{S}_{\text{sim}}$ is the the sampled state space from the $\epsilon$-greedy simulation.

### 4.4.5 A Rolling Horizon MILP Approach

A description of the rolling horizon MILP and the rolling horizon 2 stage stochastic program is as follows:

**Step 0-Initialize** Set $t_{start} = 0$ and system state $s_{start}$ .

**Step 1-Increment** Set $t_{start} = t_{start} + 1$ and $t_{stop} = t_{start} + h$.

**Step 2-Generate a scenario for each random variable** The scenario generation mechanism can either be to sample the Markovian probability distribution conditioned on the current information state or by finding the most likely scenario.

**Step 3-Solve the MILP for the entire horizon** $t_{start} - t_{stop}$**.** The MILP will generate a sequence of actions: $\alpha_{start}, \alpha_{start+1}, ..., \alpha_{stop}$ and the corresponding state trajectory $s_{start} \rightarrow \alpha_{start} \rightarrow s_{start+1}, \alpha_{start+1} \rightarrow, ..., \rightarrow s_{stop-1} \rightarrow \alpha_{stop-1} \rightarrow s_{stop}$.

**Step 4-State Transition.** Apply the decision $\alpha_{start}$ and generate according to the probability distribution the stochastic realization of the random variables. Using the model equations realize $s_{start+1}$

**Step 5-Termination Check** If $t_{start} > h$. Go to Step 6 Else Go to Step 2 and set $s_{start} = s_{start+1}$.

**Step 6-Termination**

We need to comment that if we decide to use the rolling horizon framework at a stochastic problem there is a chance that one may violate the systems hard constraints. For instance, assume that we predict a high demand scenario and solve the MILP based on that scenario. The solver would generate a solution which will push large quantities of inventory to the terminal tanks. If the actual demand realization is significantly less, we would incur storage infeasibility at the terminal tanks. To circumvent such fact, one can impose such constraint as soft constraints, which means they should be added as part of the objective function with an appropriate penalty.

## *4.5 Numerical Results*

This section describes the results for the BTX supply chain system for the different solution approaches. First, we develop a solution for this application, which provides the best possible decisions. Then, we discuss the results when information is realized after the flow decisions and finally we discuss the results when information is realized before the flow decisions

### 4.5.1 An Upper Bound On The Performance

The upper bound is an MILP assuming that all the values of the uncertain parameters into the future are known before the decision are made. This provides an upper bound and a measure of comparison with the RTADP and the rolling horizon MILP. This may be considered as a way to demonstrate the value of information of knowing the future exactly.

| Architectures \ Performance | Mean ± Standard Deviation | Relative Performance |
|---|---|---|
| An Upper Bound On The Performance | $1.11 * 10^9 \pm 0.38 * 10^9$ | 100% |

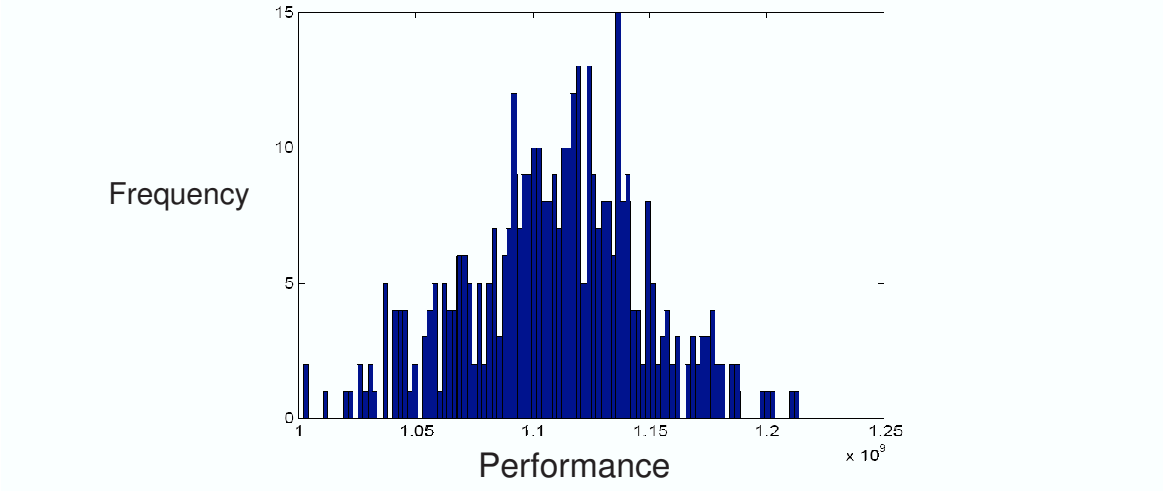



**Figure 19:** The histogram and statistics of the numerical upper bound achieved by the solution of 500 MIP with full information.

The MILP with full information is solved over 10 time periods, it was not solved for a longer horizon due an increase in solution time. Instead, we solved a full information MIP for 500 different scenarios, representing different random variable realizations. The performance mean and standard deviation as well as the histogram of these simulation are shown in Fig.19.

This represents the inherent variability in the performance of the system even when the information to compute the actions is known.

### 4.5.2 Case Study 1: Information Revealed After Mode+Flow Decisions

The cumulative results for this case study appear at Fig. 20.

| Performance / Architectures | Mean ± Standard Deviation | Relative Performance |
|---|---|---|
| An Upper Bound On The Performance | **$1.11 * 10^9 \pm 0.38 * 10^9$** | **100%** |
| Rolling Horizon MILP With Sampling From The Probability Distributions | **$0.78 * 10^9 \pm 0.06 * 10^9$** | **(70 ±5.41) %** |
| Rolling Horizon MILP With The Most Probable Scenario | **$0.79 * 10^9 \pm 0.06 * 10^9$** | **(71.2 ±5.41) %** |
| RTADP | **$0.85 * 10^9 \pm 0.08 * 10^9$** | **(76.5 ±7.2) %** |

**Figure 20:** Comparison of the tested architectures for the first case study.

The parameters used to generate the following results are identical with *Kuo and Chang* [74], and we therefore we dp not reproduced them here. The states of the random parameters follow the probability transition matrices as shown in Fig. 21 and the data of the operational mode of the Reformer unit and Tatoray unit are displayed in Fig. 22.

$$D_{c,Bz} = 10^3 \times [1.8 \quad 35 \quad 55 \quad 35] \qquad SP_{T,Bz} = 10 \times [85 \quad 85 \quad 91 \quad 91]$$
$$D_{c,Tol} = 10^3 \times [1 \quad 30 \quad 40 \quad 20] \qquad SP_{T,Tol} = 10 \times [65 \quad 65 \quad 70 \quad 70]$$
$$D_{c,Px} = 10^3 \times [8 \quad 40 \quad 50 \quad 25] \qquad SP_{T,Px} = 10 \times [68 \quad 68 \quad 72 \quad 72]$$
$$D_{c,Ox} = 10^3 \times [5 \quad 25 \quad 45 \quad 25] \qquad SP_{T,Px} = 10 \times [65 \quad 65 \quad 68 \quad 68]$$
$$D_{c,Mx} = 10^3 \times [1 \quad 1] \qquad SP_{T,Mx} = 10 \times [54 \quad 57]$$

$$P_{D_{BZ}-SP_{BZ}} = \begin{bmatrix} 0.1 & 0.6 & 0.28 & 0.02 \\ 0.06 & 0.1 & 0.6 & 0.24 \\ 0.02 & 0.08 & 0.6 & 0.3 \\ 0.8 & 0.04 & 0.06 & 0.1 \end{bmatrix}$$

$$P_{D_{BZ}-SP_{BZ}} = P_{D_{Tol}-SP_{Tol}} = P_{D_{Px}-SP_{Px}} = P_{D_{Ox}-SP_{Ox}}$$

$$P_{D_{Mx}-SP_{Mx}} = \begin{bmatrix} 0.2 & 0.8 \\ 0.9 & 0.1 \end{bmatrix}$$

**Figure 21:** Probability transition matrices and information state variables concerning the uncertain variables.

$$YD_{R,LN,Mode_{R1},Top} = [0.7 \quad 0.05 \quad 0.05 \quad 0.1 \quad 0.1]$$
$$YD_{R,LN,Mode_{R2},Top} = [0.65 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.05]$$
$$YD_{R,LN,Mode_{R3},Top} = [0.75 \quad 0.05 \quad 0.05 \quad 0.1 \quad 0.05]$$
$$YD_{R,LN,Mode_{R4},Top} = [0.6 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1]$$
$$YD_{R,LN,Mode_{R5},Top} = [0.75 \quad 0.05 \quad 0.05 \quad 0.1 \quad 0.05]$$
$$RF_{Mode_{T1},\{C_9,Tol\}} = [1 \quad 0]$$
$$RF_{Mode_{T2},\{C_9,Tol\}} = [0.67 \quad 0.33]$$
$$RF_{Mode_{T3},\{C_9,Tol\}} = [0.5 \quad 0.5]$$

**Figure 22:** Data concerning the operational modes of the Reformer unit and Tatoray unit.

#### *4.5.2.1 Setting Parameters For The Implementation of RTADP*

The number of episodes was set to 10,000 states and each simulation trial was executed with a horizon length of $h = 10$ time periods. A discount factor ($\gamma = 0.9$) was used, which makes the impact of any states beyond 10 future time periods negligible.

The discrete event simulation of the supply chain system starts given a value table with one entry. At each trial one can choose as starting state any state within the table. For convenience, each trial starts with the same $s_t$.

To construct the $A_{sub}$ at each loop we include: 10 random actions, 10 sample path MILP actions, 10 random actions by perturbing each sample path MILP action action, a best known action, when it is available and the $k$ best stored actions from the $k$-nearest neighbors of $s_t$ ($k = 4$). Specifically, each action leads to 512 [3] successive realizations, each of which has a certain probability.

Next, we describe the parameters $\delta$ and $W$ that define the neighborhood of $s_t$. The Euclidean measure $\delta$ demarcates the "neighborhood" of a particular $s_t$.

A neighboring state $s_N$, with respect to $s_t$ has the following characteristics:

**Characteristic 1:** The state dimensions of $s_N$ which express the tank levels $Y_{u,s}$ and $Y_{u,s}$ should be within 100 gallons of the corresponding tank levels of $s_t$.

**Characteristic 2:** The information state dimensions of $s_N$, which are $\hat{SP}_{c,p}(t)$ and $\hat{D}_{c,p}(t)$, should have the same realization of the random variables of $s_t$.

The value table can be filtered to find states that satisfy the above two characteristic. First, we define the feature weighting diagonal matrix $W$ by weighing equally each of the tank levels. Then, we can quantify the value of $\delta$ from Eq.(59), which corresponds to the threshold that determines what state is considered neighboring or not.

#### *4.5.2.2 The Implementation of The Rolling Horizon MILP*

Dynamic programming algorithms are designed to address the multistage nature of such problems. We apply the myopic rolling horizon MILP to the same instance for the same 500 scenarios.

As we can see from the results, the importance of the multistage stage uncertainty can be captured from the difference in performance between the RTADP and the rolling horizon MILP, which is $\simeq$6% for 10 time periods. The performance gap between these two methods is not very large, because the RTADP mainly utilized the numerical actions suggested by the MILP. Assuming that our system is at state $s_t$, the actions generated by the MILP's are the ones that generate the most profit for the system. These actions are evaluated against an expectation and a maximization operator. At the end of the RTADP routine the actions

[3] The successive states are 512, since there are 5 random variables. The first 4 have 4 states, while the $5^{th}$ has only 2 states.

selected for each state are the ones that correspond to the largest on expectation myopic reward.

### 4.5.3 Case Study 2: Information Revealed After The Mode And Before The Flow Decisions

The results for this case study are summarized at Figure 23.

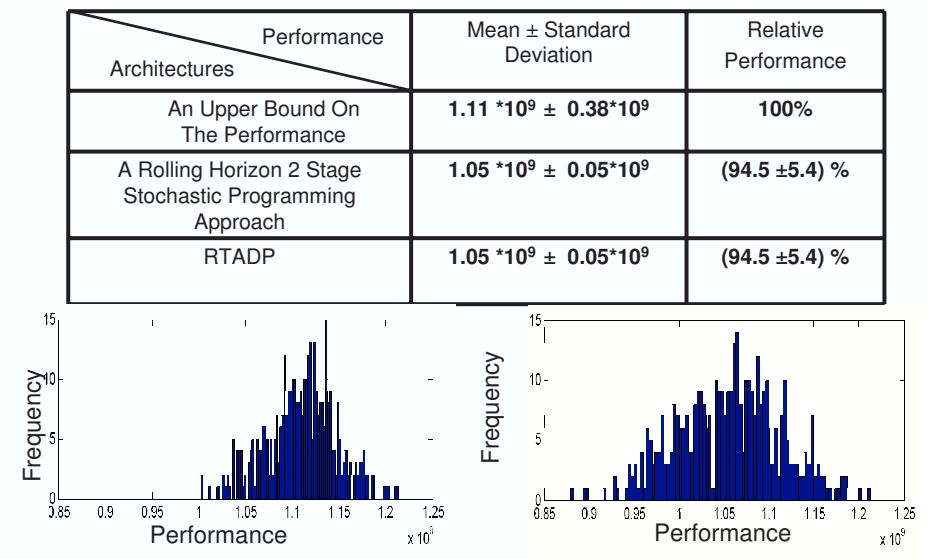



**Figure 23:** Comparison of the tested architectures for the second case study. On the left we display the histogram that corresponds to the upper bound on the performance, while on the right we display the histogram derived from both RTADP and 2 stage rolling horizon approach. We remind to the reader that these architectures were tested on the same 500 scenarios.

Figure 24 illustrates the implementation of the 2 stage stochastic programming approach within the same time period , on a problem where the decision space has two different time scales. The mode decision is common across all the realizations of the random variables. The simulation using the 2 stage stochastic programming is as follows: For periods $t = 0$ until $t = 10$ a) the system is at state $s_t$, b) run the 2 stage stochastic programming formulation and derive the solution, c) realize the random variables from the probability distribution, d) read the 2 stage stochastic programming solution and apply the transition-model equation and realize state $s_{t+1}$ e) set $s_{t+1} = s_t$ and return to a).

The performance gap compared to the solution with full information is only 5.5%. This

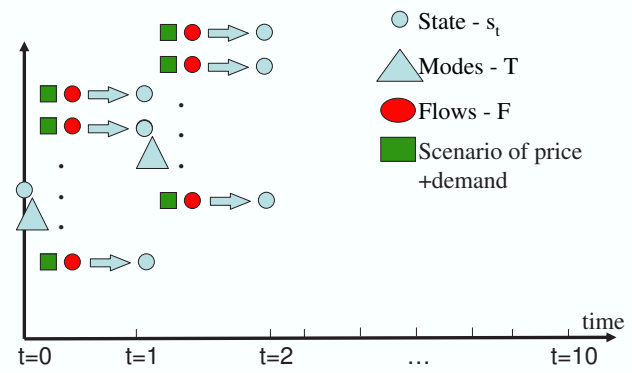



**Figure 24:** Schematic implementation of the rolling horizon 2 stage stochastic programming approach within the same time period.

demonstrates that the flow decision are able to cope with the decisions and that the mode decisions are not a significant constraint on the flow decisions.

The RTADP approach yields the same results as the rolling 2 stage stochastic programming approach. This is because the RTADP approach chooses the same actions as the 2 stage stochastic programming for the visited states and does not find any significantly better actions. T

### 4.5.4 The Value Of Information

The overall results are summarized in Fig.25. This enables us to evaluate the relative timing of information and decisions. The timing of the mode decisions does not have a significant effect in this case. Similarly making flow decisions before the uncertain parameters are known is significant. The improvement that can be made with a dynamic programming approach also depends on the performance gap. More improvement over a rolling horizon approach is possible when the timing of the decision matters. The gap between the RTADP solution and the full information solution gives a quantitative bound on the value of knowing the demands and prices before the decisions have to be made. The gap between the RTADP and the rolling horizon solutions gives the value of being able to better anticipate the impact of the value of the future on current decisions.

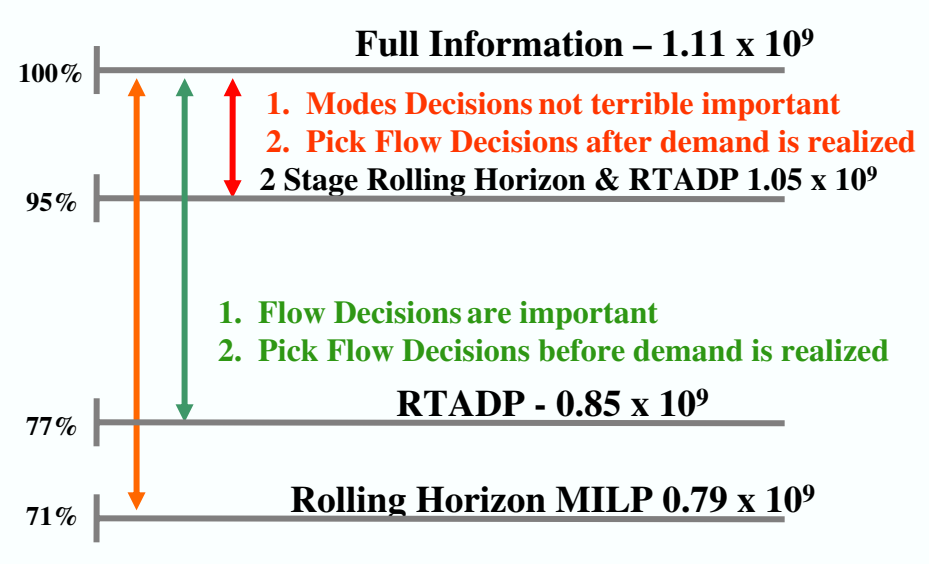



**Figure 25:** Summary - Value of Information .

## *4.6 Conclusions*

When one wants to address multistage stochastic problems, the main tools are DP and stochastic mathematical programming. In this chapter, we developed an approximate dynamic programming approach for a BTX supply chain problem. We compared this to a rolling horizon, which is one of several was to employ mathematical programming in this context (*Balasubramanian and Grossmann*) [80, 81].

The real time approximate dynamic programming approach as presented here is attractive, because one can superimpose it on a range of decision making methodologies from complex and computationally demanding deterministic mathematical programming or simple heuristics or even exploratory randomized actions. It is not clear how much effort to devote to developing good initial action and state trajectories versus allowing random exploration to find them. The proposed iterative methodology promises to choose the greedy control at each time period and to not discard valuable information about how that choice plays out in the future. Moreover, because this methodology is iterative and is based on asynchronous value iteration it can improve online performance as the number of iterations increase.

# CHAPTER 5

# CONTROLLED EXPLORATION OF THE STATE SPACE VIA AN OFF-LINE ADP APPROACH

This chapter addresses the problem of finding a control policy that drives a generic discrete event stochastic system from an initial state to a set of goal states with a specified probability. The control policy is iteratively constructed via approximate dynamic programming (ADP) over a small subset of the state space that is evolved via Monte Carlo simulations with the iteration. Algorithmic details of the approach are delineated and the effects of certain user-chosen parameters of the algorithm are investigated. The method is evaluated on several stochastic shortest path (SSP) examples and a manufacturing job shop problem introduced in chapter 3. In order to illustrate the scaling of computational and memory benefits with respect to the problem size, we solve SSP problems up to one million states. In the case of the manufacturing job shop example, we compare the performance of the proposed ADP approach with that of the rolling horizon math programming approach.

## *5.1 Introduction*

Finding the shortest path in a graph or a network is a classic optimization problem. Examples of chemical and industrial engineering problems that have been formulated as shortest path problems are: 1) logistic problems that involve the transportation of hazardous materials [82], 2) fast shortest path computation to GPS terminal enabled vehicles [83], and 3) biological problems that involve parent crossover for protein generation [84]. For deterministic versions of SSPs, a well-known solution is the Dijkstra's algorithm [85] with complexity $O(n\log n)$ where $n$ is the number of nodes at the graph is well known. Another graph search algorithm that uses a heuristic to guide its choice of paths is the A* algorithm [86]. However, neither of these algorithms can efficiently handle negative cost arcs or uncertainty in the form of stochastic state transitions, where an action one chooses at a node may not lead

to just one node but to a set of nodes based on a probability distribution over the set.

In this context, let us consider a two dimensional SSP in which each node on the grid corresponds to a state of the environment. In each state, the agent can choose a movement along one of the four compass directions are possible. However, such a decision may not necessarily move the agent along the intended direction but any of the four possible directions. The actual realized movement will be decided in accordance with the probability of **p** in the intended direction and with probability (1-**p**)/ 3 in each of the remaining three directions. Naturally, the closer $p$ is to 1, the more deterministic the problem becomes and vice versa. Actions that would take the agent off the grid leave its location unchanged without any penalty. A cost is incurred based on which state is visited. Finally the objective is to find the minimum expected cost path from a starting state to one of the goal states. A direct generalization to the SSP problem would be to allow more than four directions for movement from each system state. Again, each intended movement will guide the system to a specific set of states with corresponding probabilities.

Such a problem falls under the class of multistage stochastic decision problems, which can be solved either by stochastic programming (SP) or dynamic programming (DP). The exact SP formulation of the multi-stage problem generally yields an intractable problem even for the seemingly simple SSP. For the particular 2-d SSP problem discussed above, the number of branches in the scenario tree is $4^T$, where $T$ is the horizon length. Note that, in this problem, $T$ itself is a random variable. A reasonable heuristic solution approach for such a multi-stage problem is a rolling 2-stage SP, in which a two-stage SP is solved at each decision instance after the transition is realized. The two-stage SP, which must be solved on-line, can be handled either by an efficient sample average approximation algorithm [21] or by Benders or an equivalent Langragean decomposition.

Alternatively, DP can be used. Usefulness of DP is compromised by several computational obstacles collectively termed as the "curse of dimensionality (COD)". The COD of DP has motivated the development of an approach called Approximate Dynamic Programming (ADP), which attempts to derive an approximate solution (*i.e.*, near-optimal solution)

by using simulation and function approximation, under the formalism of dynamic programming. For a thorough literature review of ADP, we refer the reader to [87],[1]. The main thought behind ADP is to minimize the effect of the intractably large state space (to be denoted by $\mathbb{S}$) by intelligently sampling the state space and then iteratively building a value function approximation (or value table) through Bellman iteration and function approximation. A naive approach to sampling the state space is to employ a grid of uniformly spaced samples. With this scheme, however, the computation and storage grow exponentially with the dimension of the state space. Instead the sampling can be focused to relevant parts of the state space by simulating the system under the policies evolving under the iteration, starting some a priori available policies such as heuristics. This approach was tested on several process control and operations problems by [64] and [17, 18].

An alternative way to sample the state space is to utilize asynchronous dynamic programming in the form of Real Time Dynamic Programming (RTDP), which was first introduced by [15]. This approach uses estimates of the quality of states and actions to make decisions. It starts with an "optimistic" valuation of all the states and then evolves the value function of the states visited during simulation based on the Bellman's optimality equation. The drawback of this approach is that it can potentially explore the entire state space before convergence, due to the optimistic initial valuation. A modification to the RTDP algorithm has been recently proposed by [88] to address this difficulty. The approach can control the degree of exploration and hence the explosion of the size of the value table. It proposes to start with a rather pessimistic value function valuation for all the states to restrict the exploration of the discrete state space and then to use a non-parametric value function approximator and an "adaptive action set," which keeps candidate actions for each registered state to resolve the COD with respect to the action space. The downside is that convergence to the optimal policy cannot be guaranteed under this scheme. Recently, there have been several variations of the classic RTDP that improve the rate of convergence. One such variation is the Focused Real Time Dynamic Programming proposed by [89]. This approach focuses the computation selectively on system states based on the estimates of their quality and the uncertainty surrounding that quality.

The motivation for this chapter is to provide a quantitative method to explore and identify a relevant subset of states for stochastic shortest path (SSP) problems with random transitions. The main contributions of this work are: a) an ADP algorithm equipped with a structured exploration scheme for SSP problems, b) evaluation of its computational behavior with respect to problem and algorithm parameters; c) demonstration of the use of the approach for a realistic queuing network reformulated as a SSP, and d) comparison of the ADP approach with a traditional mathematical programming based strategy.

The proposed approach begins by solving the corresponding deterministic SP problem, in which movements are assumed to be completely deterministic (corresponding to the probability $\mathbf{p}= 1$). Then the algorithm is initialized by recording the corresponding values and actions for the visited states. Note that we only begin with a "partial" policy, meaning decisions are recorded only for those states on the optimal trajectory for the deterministic SP problem. Then the entries are added and the values and actions for the registered states are improved respectively by iterating between Monte Carlo simulations under the given partial policy and Approximate Value Iteration (AVI), as proposed by [64]. This iterative approach is terminated when the frequency of visiting states not registered in the table drops down to a negligible level, *i.e.*, almost all MC simulations stay in the subset of the states registered until ending up in the goal state(s). The output of this procedure is a set of sampled states and their valuation, which is converged within a given tolerance. This in turn defines a policy that gives an action for any encountered state.

Within the proposed approach, the key user-chosen parameters are: a) the number of MC simulations performed for each iteration and b) the value iteration tolerance. Those parameters dictate the computational overhead of the approach and the exploration rate. We will illustrate the role of these parameters through several simulation exercises.

This chapter is organized as follows. In section 5.2, the SSP we study is defined in precise mathematical terms. In section 5.3, the proposed approach is delineated in depth. Simulation results obtained for several SSP problems are presented in section 5.4, where key insights learned are also highlighted. In section 5.5, a realistic queuing network example is formulated as a SSP problem and solved. Finally in section 5.6, we summarize our

contributions.

## 5.2 Statement of SSP Problem

The formal mathematical statement of the SSP problem is formulated at section 2.8.

## 5.3 Overall Structure Of The Approach

As discussed at the introduction, the main idea of this quantitative approach is to focus the computation on a small portion of the state space. To achieve this we seek to expand the state entries in the value table gradually as the policy evolves in an iterative manner. The set of states in the value table at iteration $i$ will be denoted by $\mathbb{S}_i$. The iteration is terminated when a certain termination criterion is met.

The overall approach is depicted in Figure 26,27. After the initialization, one iterates between Monte Carlo Expansion (MCE), which adds entries to the value table through $N$ simulation runs, and Approximate Value Iteration (AVI), which updates the values and hence the policy by iterating on the Bellman's optimality equation. This iteration between MCE and AVI is terminated when the frequency of encountering states not registered in the value table before reaching the goal state drops down to a negligible level (set by the user).

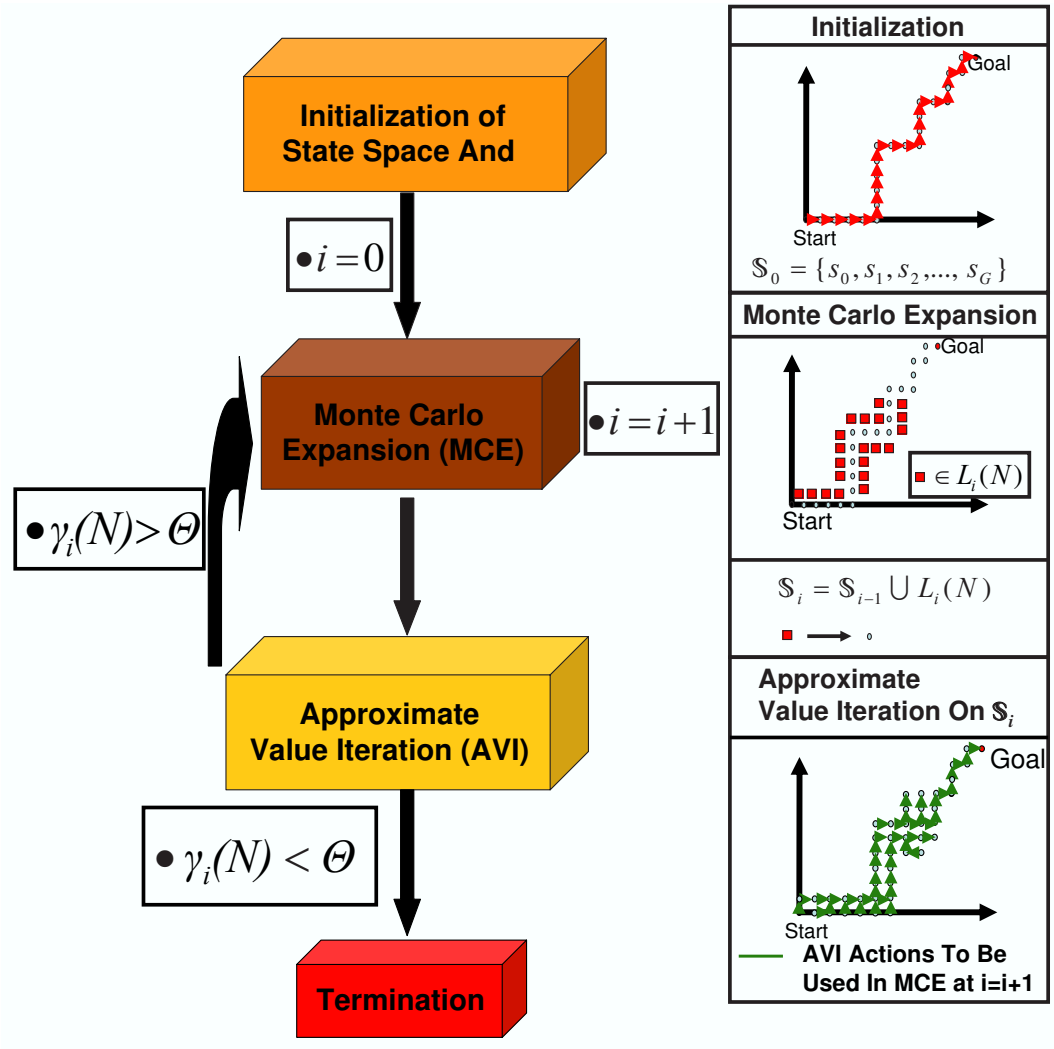



**Figure 26:** A simplified version of the structure of the proposed off-line ADP approach.

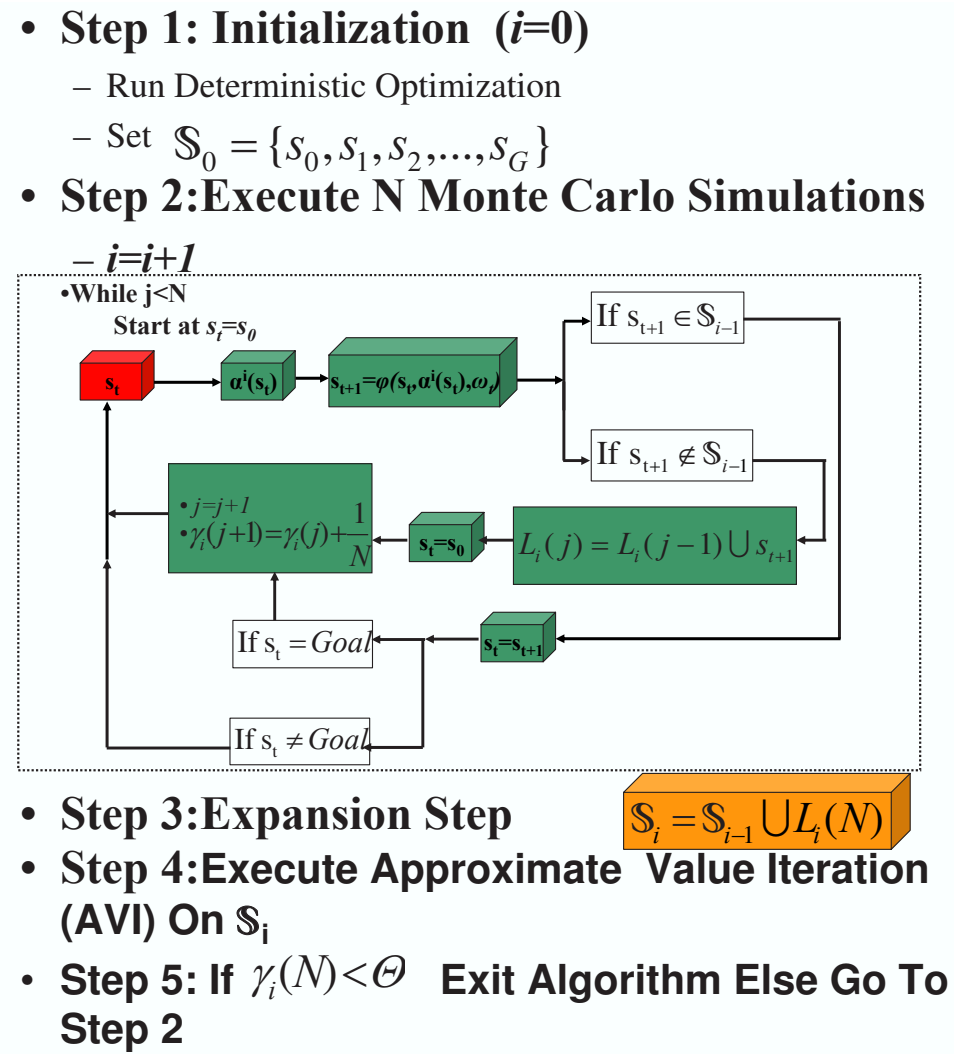



**Figure 27:** A more detailed version of the overall structure of the proposed the proposed off-line ADP approach.

### 5.3.1 Initialization

For best results, the initialization procedure should be tailored to the specific application. A general procedure we propose is to use deterministic mathematical programming after relaxing the exogenous uncertainty. In this case, the sampled states are restricted to the system states sampled along the derived deterministic mathematical programming action trace. The initialization corresponds to iteration index $i = 0$ and the corresponding set of states collected is therefore denoted by $\mathbb{S}_0$.

### 5.3.2 Monte Carlo Expansion

This phase appends additional states to the existing value table. Added entries correspond to the new states that are visited during $N$ runs of MC simulations, where $N$ is a user-chosen parameter. These states form an evolving set denoted as $L_i(j)$ where $j = 1, \cdots, N$ is an index for the MC runs. At the end of $N$ runs, $\mathbb{S}_{i-1}$ entered in the value table at iteration $i-1$ is expanded into $\mathbb{S}_i$ by adding the set $L_i(N)$.

Note that $j^{\text{th}}$ MC simulation run ends as soon as a state outside $\mathbb{S}_{i-1} \cup L_i(j-1)$ is encountered. The inputs to the MC-Expansion subroutine are: a)the number of simulations $N$, b)the starting state $s_0$, c)a set of goal states denoted as $\mathbb{S}_G$. The goal set can be a singleton or can be composed from more than one states. A formal description of the MC expansion step follows:

**Subroutine Name: MC-Expansion**

**Step 1** Initialize $j = 1$, $t = 0$ , $\gamma_i(0)$=0, $s_t = s_0$,$L_i(0) = \emptyset$ and $\gamma_i(0) = 0$.

**Step 2** **While** $j \leq N$,

Perform one of the following three steps:

**Step 2.1** For $s_t$, find the action $\alpha_t^i(s_t)$. Then transition to $s_{t+1} = \varphi(s_t, \alpha_t^i(s_t), \omega_t)$.

- If $i = 0$ the actions correspond to the ones produced by the initialization procedure, in this case they correspond to the mathematical programming action trace.
- If $i \neq 0$ the actions correspond to the ones produced by the Approximate Value Iteration procedure as explained in paragraph 5.3.3.

**Step 2.2** **If** $s_{t+1} \in \{(\mathbb{S}_{i-1} \cup L_i(j)\} \setminus \mathbb{S}_G$.

Set $s_t = s_{t+1}$ and return to Step 2.1.

**End If**

**Else If** $s_{t+1} \notin \mathbb{S}_{i-1} \cup L_i(j)$.

Add $s_{t+1}$ to the expansion list $L_i$, if it is not already registered, $\rightarrow L_i(j) = L_i(j-1) \cup s_{t+1}$

Set $s_t = s_0$

Update $\gamma_i(j) = \gamma_i(j-1) + \frac{1}{N}$

Update $j = j + 1$

Return to Step 2.

**End If**

**Else If** $s_{t+1} \in \mathbb{S}_G$.

Set $s_t = s_0$

Update $\gamma_i(j) = \gamma_i(j-1)$.

Update $j = j + 1$.

Return to Step 2.

**End If**

**End While**

**Step 3** Add to $\mathbb{S}_i$ the states within the expansion list, $\mathbb{S}_i = \mathbb{S}_{i-1} \cup L_i(N)$. $\gamma_i = \gamma_i(N)$.The initialization of the value function can be arbitrary [1], since the Approximate Value Iteration that follows consists of a contraction mapping.

The output of the MC-Expansion subroutine at iteration $i$ is: a) the updated $\mathbb{S}_i$, b) a measure $\gamma_i(N)$ that quantifies how often the existing policy visits states outside the $\mathbb{S}_i$ as a fraction of $N$ trials.

### 5.3.3 Approximate Value Iteration.

After the MC expansion phase, the approach executes an Approximate Value Iteration (AVI), a procedure similar to the one used by [64], and updates the best known policy. This part of the algorithm assumes that the value table contains a number of collected system states and their value function. The AVI algorithm, as presented here, defines a linear contraction mapping and therefore is guaranteed to converge [64].

**Subroutine Name: AVI**

**Initialize** the value function $J^0(s_t), \forall s_t \in \mathbb{S}_i$

[1] For our numerical experiments we used the value of zero.

**Step 1** For each $s_t \in \mathbb{S}_i$

**Step 2** Evaluate all actions in $\mathbb{A}$ for $s_t$ using the Bellman equation Eq.(58)

$$\begin{aligned} J^{j+1}(s_t) = \quad & \min_{\alpha \in \mathbb{A}} \mathbb{E}\{f(s_{t+1}, \alpha)\} + \\ & \gamma \sum_{s_{t+1} \in \mathbb{S}} P(s_{t+1}|s_t, \alpha) J^j(s_{t+1}) \end{aligned} \tag{58}$$

$j$ : value iteration index. Note that the entire state space $\mathbb{S}_i$is swept through in each iteration.

**Step 3** Terminate when,

$\|J^{j+1}(s_t) - J^j(s_t)\|_\infty < e_{VI}, \forall s_t \in \mathbb{S}_i \subseteq \mathbb{S}$

The output of the AVI subroutine is an updated decision rule for each state that belongs to the updated value table. The decision rule for a given state is given according to Eq.(59).

$$\begin{aligned} \alpha^*(s_t) = \quad & \mathbf{arg}\min_{\alpha \in \mathbb{A}} \mathbb{E}\{f(s_{t+1}, \alpha)\} + \\ & \gamma \sum_{s_{t+1} \in \mathbb{S}} P(s_{t+1}|s_t, \alpha) J^*(s_{t+1}) \end{aligned} \tag{59}$$

The execution of the AVI may require information about system states that are not registered in the value table. In general, we will encounter one of the following scenarios during each value function update:

**Scenario 1:** All $s_{t+1}$'s have values registered in the value table. We use these values to calculate $J^j(s_t)$.

**Scenario 2:** Some of the $s_{t+1}$'s are not found in the value table. Depending on the application, we treat this scenario accordingly.

**For The Stochastic Shortest Path Problem:** In order to use the Bellman equation, we first define a set $M_1$ and a set $M_2$ with the following properties. $M_1 = \{s_{t+1} = \Phi(s_t, \cdot, \cdot) : s_{t+1} \in \mathbb{S}_i \text{ and } P(s_{t+1}|s_t, \cdot) = 0\}$, $M_2 = \{s_{t+1} = \Phi(s_t, \cdot, \cdot) : s_{t+1} \notin \mathbb{S}_i \text{ and } P(s_{t+1}|s_t, \cdot) \neq 0\}$. The cumulative probability TP, from a state $s_t$ under a specific action $\alpha_t$, of not visiting a registered to the value table state is:

$$TP(s_t, \alpha_t) = \sum_{k_1=1}^{|M_2|} P(s_{k_1}|s_t, \alpha_t), \forall s_{k_1} \in M_2 \tag{60}$$

In order to utilize Eq.(60), the probabilities for each of the states that belong to set $M_1$ and $M_2$ are readjusted as follows.

$$P(s_{t+1}|s_t,\alpha_t) = P(s_{t+1}|s_t,\alpha_t) + \frac{TP(s_t,\alpha_t)}{|M_1|}, \forall s_{t+1} \in M_1 \tag{61}$$

$$P(s_{t+1}|s_t,\alpha_t) = 0, \forall s_j \in M_2 \tag{62}$$

**For The Queuing Problem:** This case is effectively covered in chapter 3,4. For completeness of this chapter, we need define the details of the $k$ nearest neighbor non parametric approximator that is used for the numerical results.

For this scenario, we need to find the set of entries within $\delta$ distance of $s_{t+1}$ (to be denoted by $\mathcal{N}_\delta(s_{t+1})$). We use the weighted Euclidean distance metric $d$, as proposed by [64], with a user-chosen parameter $\delta$:

$$\mathcal{N}_\delta(s_{t+1}) \stackrel{\text{def}}{=} \left\{ s \in S \ : \ d = \sqrt{(s-s_{t+1})^T Z (s-s_{t+1})} < \delta \right\} \tag{63}$$

In the above, $Z$ is a feature weighting diagonal matrix. If $|\mathcal{N}_\delta(s_{t+1})| \geq k$, we approximate the value function of $s_{t+1}$ from the $k$ nearest states recorded in the value table, by taking an average, as follows:

$$J^j(s_{t+1}) = \frac{1}{k} \sum_{x \in \mathcal{N}_k(s_{t+1})} J^j(x) \tag{64}$$

where $\mathcal{N}_k(s_{t+1})$ denotes the set containing the k nearest neighbors. In the case that $s_j$ has $k' < k = 4$ neighbor states within the specified distance, we use Eq.64 with just $k'$ states to approximate $J^\pi(s_j)$.

**Scenario 3:** The case where $|\mathcal{N}_\delta(s_{t+1})| = 0$ is omitted, since the way that the approach is designed there will always exist at least one neighbor state to $s_{t+1}$.

*Lee et al* [64] has proved analytically that synchronous VI with the usage of $k-$NN consists a contraction mapping that will uniformingly converge the value function for the sparsely sampled states. That converged function will not be the optimal value function, since they eclectically sample the state space.

An intuitive interpretation that explains the reason that this scheme will result to convergence follows: Assume that for each state the value function backups fall under scenario 2. The end result of the $k-$NN methodology will be to redistribute the probabilities of visiting $s_{t+1}$'s. That by itself will define a different probability transition matrix than the original one. This different probability transition matrix naturally will result to a different fixed point solution.

Therefore an interesting quantitative question that one can ask is: how much can the original entries of the probability transition be perturbed, in order to guarantee that the $k-$NN converged value function will instruct a policy close to be optimal ? To the best of our knowledge, one cannot give a quantitative answer to such a generic question. The only way to answer something like this is via extensive simulations and is highly problem and parameter dependent.

### 5.3.4 Termination Criteria

The iteration between the expansion phase and the AVI phase is terminated, when the frequency of visiting, states outside the value table denoted as $\gamma_i(N)$ drops below a given threshold $\Theta$. Inevitably in the early stage of the algorithm, the current partial policy will visit system states outside the existing value table, due to the random outcome of the actions. We define "frequency" as an estimate of the probability of visiting states outside the existing value table with the existing best known policy. To evaluate the frequency, we execute $N$ Monte Carlo (MC) simulations given this policy . Note, that each MC run $j$ starts from the starting state and is terminated after we visit a $s_t \notin \mathbb{S}_{i-1}$ or the goal state $s_G$. At each simulation that we encounter a $s_t \notin \mathbb{S}_{i-1}$ , the frequency $\gamma_i(j)$ is increased by the quantity $\frac{1}{N}$. $(\gamma_i(1) = 0 \quad \forall \quad i)$.

$$\gamma_i(j+1) = \gamma_i(j) + \frac{1}{N} \tag{65}$$

At any stage of the algorithm, the frequency can be used as a measure of our lack of knowledge of the performance of our current policy

## *5.4 Numerical Results On The Shortest Path Problem*

The computational overhead and the overall performance achieved by the algorithm is dependent on its tuning parameters. In this section, we will optimize the usage of those parameters when used by our approach by performing several numerical experiments.

### 5.4.1 Quantitative Selection Of Tuning Parameters

The exploration of the proposed approach is influenced by the following parameters: a) the number of simulations $N$, performed at the MC expansion step, b) the value iteration tolerance parameter $e_{VI}$, which determines the computational load of the AVI step as well as the quality of the actions that guide the value table expansion via the MC simulations and c) the termination threshold parameter.

#### *5.4.1.1 On Varying N*

This numerical example is on a 30x30 grid with a) **p**=0.8, b) $e_{VI}$=1 and c) $\Theta = 0.1$. The cost structure is posted at http://www.chbe.gatech.edu/lee/members /npratikakis.shtml under section 4.1.

To examine the behavior of the approach with respect to parameter $N$, we set the following experiments: 1) $N$ =100, 2) $N$ =200 and 3)$N$ =500.

The outcome of the simulations is shown at Table 9. As $N$ increases, the number of the algorithmic iterations decreases. By the term 'iteration', we mean the MC expansion followed by the AVI procedure.

**Table 9:** The influence of the parameter N on the achieved exploration rate and the number of iterations. .

| Number of MC Simulations | Number of states connected | Number of iterations |
|---|---|---|
| 100 | 363 out of 900 | 30 |
| 200 | 368 out of 900 | 23 |
| 500 | 376 out of 900 | 17 |

#### 5.4.1.2 On Varying $e_{VI}$

For the same cost structure as before and by setting $N = 100$, we set $e_{VI}$ at 1 and at 5 respectively. If $e_{VI}$ is set to 5, then the AVI does not fully converge and the resulting partial policy is somewhat random suggesting free exploration of the state space. If $e_{VI}$ is set to 1, we sufficiently converge the value functions at each iteration achieving directed and minimum exploration. (The results as appear at Table 9 are consistent, across all tested instances.) By the term 'iteration', we mean the MC expansion followed by the AVI

**Table 10:** The influence of the AVI tolerance parameter on the achieved exploration rate.

| AVI Tolerance$e_{VI}$ | Number of states connected | Number of iterations |
|---|---|---|
| 1 | 363 out of 900 | 30 |
| 5 | 868 out of 900 | 126 |

procedure.

In conclusion, for this example, in order for the technique to direct and restrict the exploration to a small fraction of the state space and thereby to be computationally efficient, a high value of $N$ MC simulations should be instructed as well as a low value of $e_{VI}$.

#### 5.4.1.3 On Varying The Termination Criteria

At this paragraph we consider a different example, in order to test how the termination criterion affects the exploration behavior. Its cost structure can be retrieved from the webpage under section 4.1 example 2. Moreover, the starting state is set as (0,1) and the goal (0,30).

The termination threshold $\Theta$ varies from 0 to 0.05 to 0.1. As expected, the higher the threshold, the lower the exploration, but the final policy has a higher probability of visiting unregistered states. The results for example 2 are demonstrated in Fig. 28.

In conclusion, we need to set a small value of $\Theta$, in order for the technique to ensure that given the retrieved partial policy with probability $1 - \Theta$ we will not explore other states.

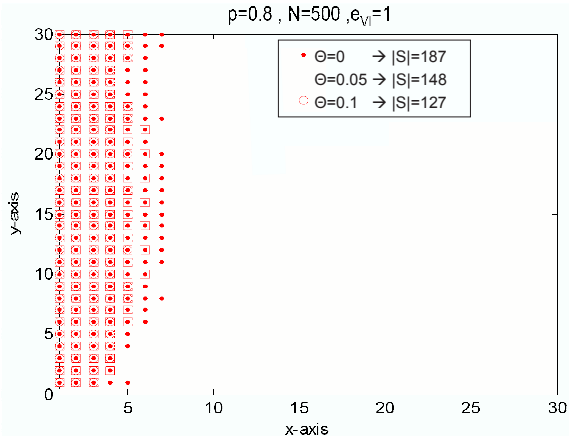



**Figure 28:** Explored states with respect to the imposed termination threshold Θ.

#### *5.4.1.4 On Varying* ***p***

The exploration instructed by our approach when changing parameter **p** is displayed at Fig.29. By decreasing variable **p**, we increase the noise level. Naturally this forces our approach to explore a larger space.

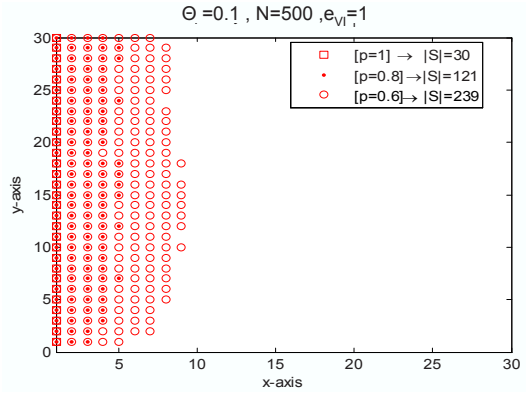



**Figure 29:** Explored states achieved by the approach with respect to the noise level **p**.

### 5.4.2 Scaling And Memory Requirement

The proposed procedure is tested on: 1) a 5x5 graph, 2) a 30x30 graph , 3) a 100x100 graph , 4) a 300x300 graph and at 5) a 1000x1000 graph, in order to capture its computational scaling and memory effect. To address the $4^{th}$ and $5^{th}$ problem, we employ the initialization procedure as explained at Appendix B. Briefly the benefits of that procedure

are: 1) it collects a small percentage of the system states; 2) it constructs a partial policy that with probability one will visit the goal state starting from the starting state and 3) it initializes the value functions of the collected states so that it would require a small number of AVI iterations for convergence. The cost structure for each of these problems is posted

**Table 11:** Problem size - Exploration rate - Reduced memory requirements with respect to the full problem.

| Problem Size Total Number of States | Number of states connected | Reduced Memory Requirement |
|---|---|---|
| 25 | 21 | 16% |
| 900 | 367 | 59% |
| 10000 | 1506 | 84% |
| 90000 | 3409 | 96% |
| 1000000 | 8803 | 99% |

online. Note, that the achieved exploration is cost structure specific. Nonetheless, our examples demonstrate that the ratio of system states collected to the problem size is steadily decreasing when moving to larger problems. From our numerical experiments, for a one dimensional SSP problem which enumerates 1 million states we end up collecting approximately 1% of the whole state space. By performing the AVI only to this small subset of the state space, significant computational savings are achieved (i.e., in this case two orders of magnitudes). Moreover, the memory requirements are greatly reduced, since we only need to store and perform the VI to a small sized value table.

### 5.4.3 Comparing RTADP And Off-line ADP On Shortest Path Problems

In this section, we will be comparing the RTADP algorithm with the Off-line ADP approach on the three stochastic shortest path instances (77,900 and 10,000 discrete states) that were introduced at the third chapter.

#### *5.4.3.1 Comparison Results On A 77 Discrete State Space Example*

Table 12 summarizes the results of the best of our RTADP runs and the off-line ADP approach where $N = 500$ and $\Theta = 0$ against the full DP solution.

Both algorithm perform optimally, but with the off-fine ADP we retrieve a stationary policy over a compact set of states. In other words we have the guarantee that the retrieved

**Table 12:** Comparing the best of RTADP runs with the Off-line ADP algorithm on the 77 discrete state space example.

| Algorithm | Online Performance (Relative to Full DP Percentage) | States Explored (Percentage of space explored) |
|---|---|---|
| Full DP | 309 | - |
| Best of RTADP | 309.31 | 51/77 (66.23%) |
| Off-line ADP | 309.02 | 54/77 (70.13%) |

**Table 13:** Comparing the best of RTADP runs with the Off-line ADP algorithm on the 900 discrete state space example .

| Algorithm | Online Performance (Relative to Full DP Percentage) | States Explored (Percentage of space explored) |
|---|---|---|
| Full DP | 237 | - |
| Best of RTADP | 246.35 | 311/900 (34.56%) |
| Off-line ADP | 245.41 | 359/900 (39.89%) |

policy while executed will not visit any other of these 54 states.

#### *5.4.3.2 Results On A 900 Discrete State Space Example*

Table 13 summarizes the results of the best of our RTADP runs and the off-line ADP approach where $N = 500$ and $\Theta = 0.01$ against the full DP solution.

Both algorithms perform close to optimality while identifying only 1/3 of the state space. The only fault of the policy produced by the RTADP algorithm is that it visits a small number of states outside the registered value table. To circumvent this problem the RTADP must rely to a heuristic policy that will lead the system to the registered state space. On the contrary by executing the off-line ADP with the given parameters the retrieved policy will visit unregistered states with a frequency of 1%. If we wanted the frequency to drop to zero we would tune the parameter $\Theta = 0$.

#### *5.4.3.3 Results On A 10,000 Discrete State Space Example*

Table 14 summarizes the results of the best of our RTADP runs and the off-line ADP approach where $N = 1500$ and $\Theta = 0.001$ against the full DP solution.

**Table 14:** Comparing the best of RTADP runs with the Off-line ADP algorithm on the 10,000 discrete state space example.

| Algorithm | Online Performance (Relative to Full DP Percentage) | States Explored (Percentage of space explored) |
|---|---|---|
| Full DP | 519 | - |
| Best of RTADP | 940 | 1,993/10,000 (19.93%) |
| Off-line ADP | 924 | 2,960/10,000 (29.60%) |

Both algorithms perform close to optimality and they identify less than 1/3 of the state space.

The portion of the state space identified given these two approaches is shown at Figure 30.

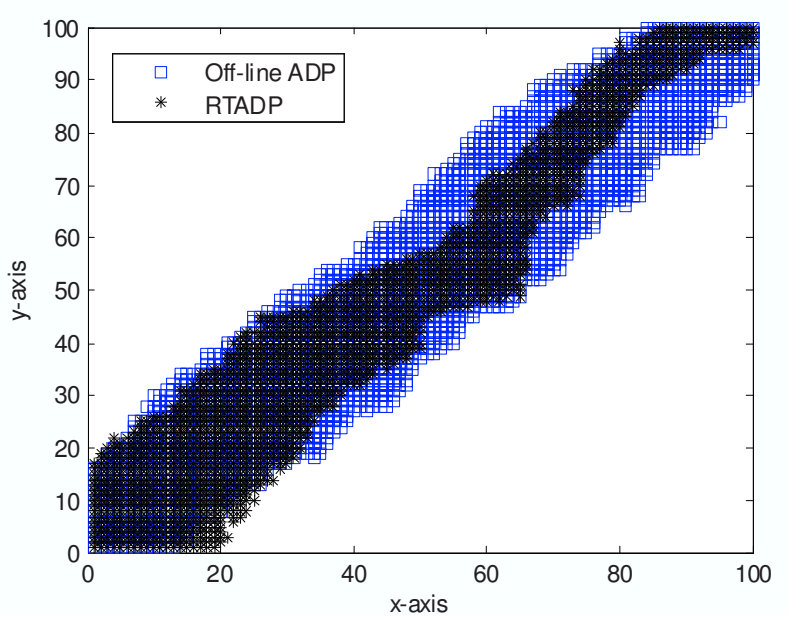



**Figure 30:** Demonstrating the porion of the state space identified by the Off-line ADP Vs the best RTADP run. The RTADP restricts the state space exploration to a subset of the states explored by the off-line ADPapproach.

A significant drawback of of the partial policy generated by the RTADP algorithm is that it still visits a states outside the registered value table. To circumvent this problem, we must run the RTADP longer and collect further states or we must rely on a heuristic policy that will lead the system to the registered state space. On the contrary by executing the

off-line ADP with the given parameters the retrieved policy will visit unregistered states with a specified frequency of 1%. If we wanted the frequency to drop to zero we would tune the parameter $\Theta = 0$ and the corresponding exploration would be greater.

## *5.5 Modeling And Results Of A High Dimensional Queuing Example*

The realistic queuing example is a modified version of the case study as studied in chapter 3.

### 5.5.1 Queuing Network Under Uncertain Demand and Product Yield

The example represents a queuing network - manufacturing process, illustrated at Fig. 6. The objective of the problem is to maintain the inventory at station 2 and 3, at a minimum level and to control the final product inventory,$I_t$, at the specified set point throughout the time horizon. The control of these buffers will be achieved via resource allocation decisions. Specifically, the manufacturing process of interest is divided into three interdependent stations with physical buffers (queues) and a final product inventory. Each station has fixed capacity. The in-process inventory queued at each stage is controlled via the simultaneous allocation of the capacity of each stage. The demand rate for the final product is stochastic and modeled as a first order Markov chain. A further complicating factor in this example is the possibility of processing failure: If a product coming out of station 2 fails to meet required specifications, the failed product is rerouted to station 3 for reprocessing. At station 3 the failed product is serviced and placed back in the queue in front of station 2. The fraction of products that meet the specs at each time period is also a random variable, which is also modeled as a first order Markov chain. The decisions at each time are the allocation of the constant capacity of each station.

The variable $W_{i,t}$ represents the in-process inventory queued at each stage $i$ at time period $t$ and the variable $I_t$ the final product inventory. The demand for the final product is a sequence of random variables $D_1, D_2, D_3, ...$ with the Markov property, meaning the future values depend solely on the present state and are independent of past states. For a first order Markov chain model, the dynamics is governed by a probability transition

matrix, whose $(i,j)^{\text{th}}$ element represents $Pr(D_{t+1} = d_i | D_t = d_j)$ where $d_i$ is the $i^{\text{th}}$ possible value for $D$. In this study, $i = 1, \cdots, 6$ and therefore the transition probability matrix $P_D$ is a $6 \times 6$ matrix. The fraction of products that fail to meet the specs at each time period is also represented by a sequence of random variables denoted by $R_1, R_2, R_3, ....$ At every time period $t$, the random variable $R_t$ may take one of two possible values. The lower state value of $r_1$ corresponds to a high product yield and conversely the higher state value of $r_2$ to a low product yield. Transition from $r_1$ to $r_2$ may be due to an unexpected event that harms the system's performance. The corresponding $2 \times 2$ probability transition matrix is denoted by $P_R$.

The formulation for this specific example as a formal MDP follows.

#### *5.5.1.1 State Vector*

For the manufacturing job shop example, the state variable is a five dimensional vector defined below:

$$s_t = \begin{bmatrix} W_{i,t} = \text{ Queue at stage } i = 2,3 \text{ at time } t \\ I_t = \text{ Finished product at time } t \\ \hat{D}_t = \text{ Realized demand value at time } t \\ \hat{R}_t = \text{ Realized recirculation rate value at time } t \end{bmatrix} \tag{66}$$

Realized values of the random variables at time $t$, $\hat{D}_t$ and $\hat{R}_t$, are assumed to be known to the decision-maker. Hence they constitute exogenous information variables. However, the values of these variables at future times are uncertain and are described by the corresponding probability distributions. It is customary to include the parameters defining these conditional probability distributions of the random variables as a part of the state vector as they capture the information available at that time period. They are therefore called information states. With a first order Markov chain model, it is sufficient to include just $\hat{D}_t$ and $\hat{R}_t$ in the system state as they completely define the conditional probability distributions of the future random variables.

For a larger queueing network, one would similarly use as a state vector the buffers as

well as the information states of the random variables.

#### *5.5.1.2 Decision Variables*

Decisions are modeled in discrete time. The decision space $\mathbb{A}$ encodes all the possible controls that are applicable to each system state $s_t$. Each action or control is concerned with: deciding the percentage of the capacity of each stage. Therefore $\mathbb{A}$ has three continuous dimensions.

$$\mathbb{A} = \left[ \; PU_{i,t+1} = \text{Capacity percentage of station } i \text{ at } t+1 \; \right] \tag{67}$$

#### *5.5.1.3 Transition Function*

The transition function is defined formally via the mass balances at the queues 2,3 and at the final inventory. Details regarding these linear equations have been discussed in chapter 3.

For chapter completeness we need to define the total number of jobs $TJ_{i,t}$ exiting station $i$ at time $t$, since this quantity appears at Fig.31. This quantity is expressed via Eq.(68):

$$TJ_{i,t} = \mu_i PU_{i,t} \quad \forall t (i = 1, 2, 3) \tag{68}$$

where $\mu_i$ is the invariant total capacity of each station $i$.

#### *5.5.1.4 Contribution (Cost) Function*

The one step cost produced by a decision $\alpha_t$ at state $s_t$ during one time period with random variable $\omega_{t+1}$ is denoted as $\hat{f}(s_t, \alpha_t, \omega_{t+1})$. With some abuse of notation we denote $\omega_{t+1} = [D_{t+1}, R_{t+1}]$, which is a two dimensional vector describing the outcome of the two independent Markovian processes at time $t+1$. These values impacts the transition from a state to a successive one. Then, the expression for $\hat{f}(s_t, a_t, \omega_{t+1})$ is:

$$\hat{f}(s_t, a_t, \omega_{t+1}) = 8 * 10^3((I_{t+1} - S_{SP})^2 + 10^3(W_{2,t+1} + W_{3,t+1}) \tag{69}$$

The expectation of the one step profit is defined over the probability space $\Omega$:

$$f(s_t, a_t) = \mathbb{E}[\hat{f}(s_t, a_t, \omega_{t+1}|s_t)] = \sum_{j=1}^{N} P(s_j|s_t, \alpha_t)\hat{f}(s_j, a_t, \omega_j) \tag{70}$$

where $P(s_j|s_t, \alpha_t)$ is the probability of $\omega_{t+1}$ taking the value of $\omega_j$ and $N$ is the number of transitions with non-zero probability starting from $s_t$.

### 5.5.2 One To One Correspondence Of The Queuing Example With The Shortest Path

Our goal is to construct a policy that will drive the system to the assigned goal states from any starting state with probability at least 1-$\Theta$.

To tailor a problem as a shortest path one needs to define: a) a set of goal states, b) the modified cost function, c) starting states and d) stochastic transitions.

For the queueing network, we define as goal states the states with the following properties: $\mathbb{S}_G = \{s_t \in \mathbb{S} : A \leq I_t \leq B \ , W_{2,t} \leq C \ , \ W_{3,t} \leq D\}$. We assign to all goal states a value function value $J(s_t) = 0, \forall s_t \in \mathbb{S}_G$ and therefore the modified cost function includes, that the single stage reward of a goal state under all possible actions is set to zero.

The starting state for the queuing network can be any system state outside the set of goal states, namely $s_0 \in \mathbb{S} \setminus \mathbb{S}_G$.

The original transition function $\varphi$ maps $s_t$ to a successive state $s_{t+1}$:
$s_{t+1} = \varphi(s_t, \alpha_t, \{\hat{D}_{t+1}, \hat{R}_{t+1}\})$. Each state transition holds a probability weight **p** that corresponds to the Markov chain joint realizations of $\{\hat{D}_{t+1}, \hat{R}_{t+1}\}$. The modified transition function adopts the original transition function and sets all the goal states to be self absorbing under all actions.

### 5.5.3 Numerical Results

The values of the parameters that are used to define the set of goal states are : A = 450 , B = 550 , C = 100 , D = 100. The goal states are effectively represented with a cube as shown at Fig.32. The rest of the problem parameters can be retrieved at Fig.31.

#### *5.5.3.1 On Varying N*

To be consistent with the previous section, we examined the behavior of the approach with respect to parameter $N$. By setting $\Theta = 0.05$, $e_{VI} = 1$ and varying $N$ as follows : 1) $N$ =50, 2) $N$ =500 and 3)$N$ =5000, the observed results appear at Table 15 .

Capacity of station 1 = 400

Capacity of station 2 = 400

Capacity of station 3 = 600

$$P_D = \begin{bmatrix} 0.8 & 0.1 & 0.08 & 0.02 & 0 & 0 \\ 0.06 & 0.7 & 0.1 & 0.07 & 0.07 & 0 \\ 0.01 & 0.08 & 0.7 & 0.1 & 0.11 & 0 \\ 0.02 & 0.02 & 0.01 & 0.8 & 0.13 & 0.02 \\ 0 & 0.01 & 0.11 & 0.13 & 0.7 & 0.05 \\ 0.02 & 0.01 & 0.01 & 0.11 & 0.15 & 0.7 \end{bmatrix}$$

$$D = [7 \quad 20 \quad 35 \quad 50 \quad 70 \quad 100]$$

$$P_R = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

$$R = [0.2 \quad 0.7]$$

**Figure 31:** Numerical values of the queuing network.

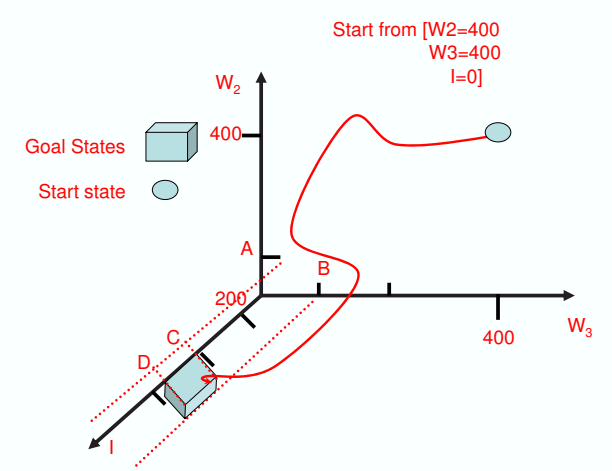



**Figure 32:** Schematic representation of a path that leads the system from an initial state to the designated goal states.

When increasing $N$, we identify more states with the same computational overhead. For $N = 5,000$ we identified twice as much states as we did for $N = 50$ , this fact didn't result to a significant impact on the performance.

#### *5.5.3.2 On Varying $e_{VI}$*

The importance of converging the value functions at each iteration is shown at the following table.

In the case where $N = 50$, when setting $e_{VI} = 100$ the output policy, given 1.000 simulation , never led the system to the goal states and therefore the average cost is $\infty$.

**Table 15:** Evaluating the algorithm's performance while varying parameter $N$.

| N | Average Performance Over 1,000 Scenarios | Iterations | $\|\mathbb{S}\|$ |
|---|---|---|---|
| 50 | 770 | 79 | 2,194 |
| 500 | 762 | 38 | 3,265 |
| 5,000 | 742 | 29 | 5,810 |

While when setting $e_{VI} = 1$, the output policy guided the system to the goal states in every simulation run.

In the case where $N = 5,000$, when choosing $e_{VI} = 100$ instead of $e_{VI} = 1$ the average incurred cost is doubled. Nonetheless, both output policies led the system to the goal states for every simulation run.

**Table 16:** Evaluating the algorithm's performance while varying parameter $e_{VI}$.

| $N$ | Average Cost Over 1,000 Scenarios | $e_{VI}$ | $\|\mathbb{S}\|$ |
|---|---|---|---|
| 50 | 770 | 1 | 2,194 |
| 50 | $\infty$ | 100 | 3,984 |
| 5,000 | 742 | 1 | 8,554 |
| 5,000 | 1501 | 100 | 5,810 |

#### *5.5.3.3 Comparing Full Information Vs ADP Vs Rolling Horizon MIQP*

The cumulative results are shown at Table 17. Those reveal that the tested algorithm com-

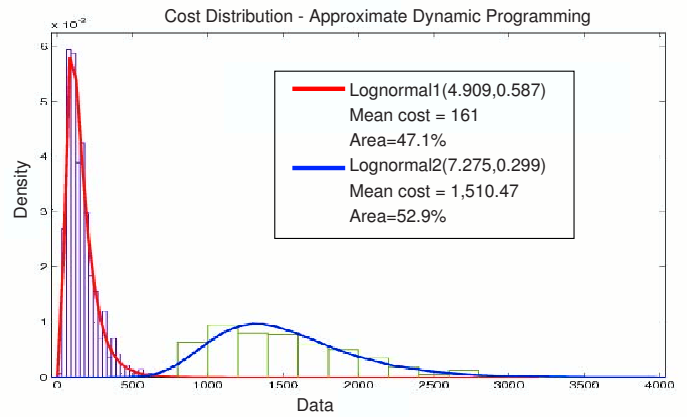
**Table 17:** Cumulative results of 3 optimization strategies on the queuing network. Each strategy is been tested on 1,000 independent sampled scenarios.

| | Full Information | ADP ($N = 500$) | Rolling MIQP |
|---|---|---|---|
| Average number of steps to reach goal states | $7.5 \pm 3.1$ | $7.5 \pm 2.8$ | $8.5 \pm 4.2$ |
| Average Total Cost | $14.7 \pm 1.5$ | $762 \pm 658$ | $1,525 \pm 1,528$ |

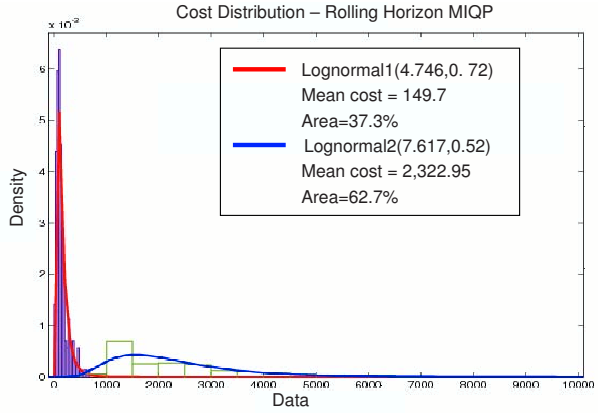
pare poorly with respect to the full information solution. Moreover, the ADP approach performs significantly better than the rolling mixed integer quadratic programming approach. The fact that the standard deviation appears to be so significant is attributed to the fact that the cost distribution have more than one modes.

The cost distributions for both architectures can be well approximated by a mixture of

two lognormal probability distribution functions(pdf). The parameters, the mean cost and the total area covered by each lognormal is shown at Fig.33(a)and Fig.33(b).




(a) Mixture of two lognormal distributions that approximate the histogram of the cost distribution when applying the ADP optimization strategy.




(b) Mixture of two lognormal distributions that approximate the histogram of the cost distribution when applying the ADP optimization strategy.

**Figure 33:** Cost distributions of ADP Vs Rolling Horizon MIQP.

Next we examine particular scenarios for the random variables and the detail implementation of the ADP Vs the rolling MIQP strategy.

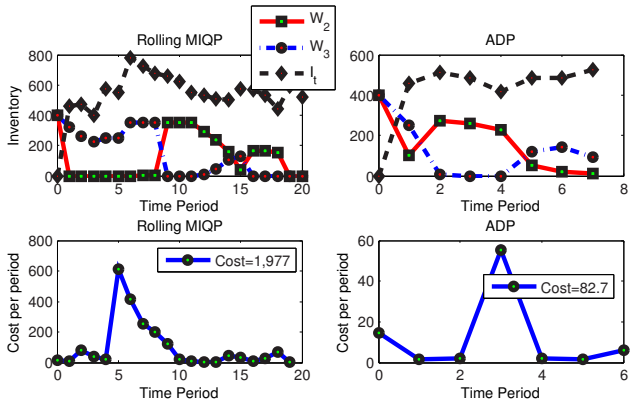
#### *5.5.3.4 On Examining Scenarios*

This paragraph examines the implementation of the rolling horizon MIQP and the ADP to two particular types of scenarios of the random variables.

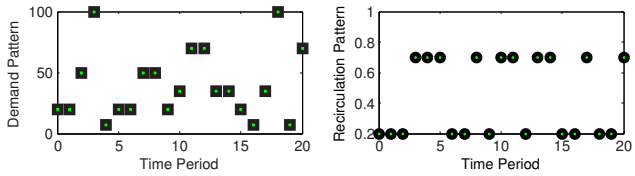
For some scenarios there is small discrepancy between the projected scenario and the one realized throughout the horizon. Therefore, the recorded cost for such scenarios is

minimum. For such scenarios both ADP and rolling horizon MIQP perform similarly .

For other scenarios there is at least once large discrepancy between the projected scenario and the one actually realized. The main problem, while using the rolling MIQP approach is the fact that the derived decisions are based on a unique sample of the random variables. Given this it may lead the system to unwanted regions of the state space. For the realized scenario as represented at Fig.34 at time period 5, the system occupies the state $s_5 = [W_{2,5} = 0 \quad W_{3,5} = 247 \quad I_5 = 572 \quad \hat{D} = 20 \quad \hat{R} = 0.7]$ and the rolling horizon MIQP instructs the action $\alpha_5 = [0.5 \quad 1 \quad 1]$, then the system transitions to state $s_6 = [W_{2,6} = 0 \quad W_{2,6} = 353 \quad I_6 = 777 \quad \hat{D} = 20 \quad \hat{R} = 0.2]$. Therefore, we incur a large penalty because $I_6$ deviated 277 units from the set point ($S_{SP} = 500$).




(a) Inventory evolution of the system and cost realization, while implementing ADP and Rolling Horizon MIQP strategy.

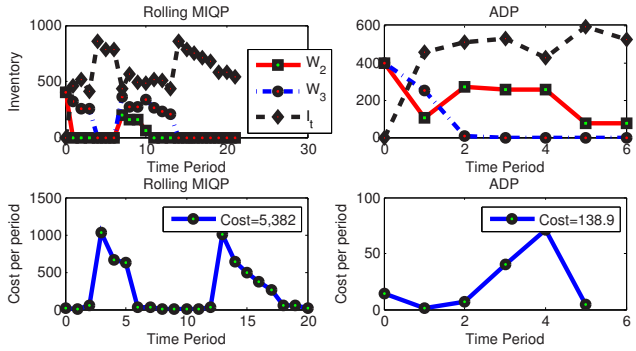



(b) Specific realization of the random variables (Demand and Recirculation pattern).

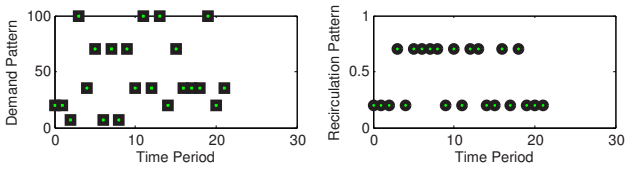
**Figure 34:** The implementation of ADP and rolling MIQP to a specific scenario.

On the contrary the actions produced by the ADP lead smoothly the system to the set of goal states . The ADP solution records a total cost of 82.7 units and the MIQP 1,877 units.

Moreover, there are the scenarios for which there are at least two large discrepancies between the projected scenario and the one actually realized as shown in Fig.35. This is the main reason that we observed two picks at the incurred cost, when we apply a rolling horizon MIQP. The ADP solution leads the system to the set of goal states in six steps and records a total cost of 138.9 units, while the MIQP records a cost of 5,382 units.




(a) Inventory evolution of the system and cost realization, while implementing ADP and Rolling Horizon MIQP strategy.




(b) Specific realization of the random variables (Demand and Recirculation pattern).

**Figure 35:** The implementation of ADP and rolling MIQP to a specific scenario.

As we can see from these results, the importance of the multistage stage uncertainty can be captured nicely from the ADP approach, but not from rolling horizon MILP.

The next subsection delineates some technical details about the implementation of the ADP as well as the rolling horizon MIQP.

#### *5.5.3.5 Setting Parameters For The Implementation of ADP*

The starting state used for this experiment is $s_0 = [400 \quad 400 \quad 0 \quad 20 \quad 0.2]$. A common discount factor ($\gamma = 0.9$) was used for all the experiments, which makes the impact to the value function of any state transition and therefore cost accumulation beyond 10 future time periods negligible.

#### 5.5.3.6 Creating An Adaptive Action Set

Since the action space is continuous we need to address the COD with respect to the action space. To achieve that we use the adaptive to the state action set , as proposed in [90]

This concept is used as follows: while executing the AVI, we constructed the adaptive action independently for each value function state update. This set of actions included: a) 50 random actions, b) 10 sample path MIQP actions , c) a best known action, when it is available and the $k$ best stored actions from the $k$-nearest neighbors of $s_t$ ($k = 4$).

We choose $Z = [0.1 \quad 0.15 \quad 0.5 \quad 4 \quad 2,000]$ such that the neighboring states share the same realization of the random variables.

#### 5.5.3.7 Formulation And Implementation of The Rolling Horizon MIQP

The MIQP formulation used for the produced results is the following:

$$\begin{aligned} &\min_{\alpha_t} \sum_{t=1}^{h} \left( \hat{f}(s_t, \alpha_t, \omega_{t+1}) \right) \\ &s.t. \\ &g(s_t, \alpha_t) = 0 \\ &\alpha_t \in \mathbb{R}^{3+} \\ &W_{i,t} \in \mathbb{Z}^{+} \end{aligned} \tag{71}$$

The horizon length $h$ for which the MIQP is solved is set to 79. By $g(s_t, \alpha_t)$ we denote the material balances that must be satisfied at each system node for consecutive time periods.

A description of the classic rolling horizon algorithm can be retrieved in [90].

## 5.6 Conclusions

For large scale problems, approximate dynamic programming has emerged as an effective way to approximate the conceptually elegant but computationally inefficient dynamic programming algorithm. The compromises made in ADP result in a tradeoff between the exploration of the state space and the exploitation of existing knowledge of the values of the already-visited states.

In this chapter, we explored these issues in the context of a stochastic shortest path problem and a queuing network with invariant capacity and inventory decisions faced with

uncertainty over the demand and performance of the system. We proposed an ADP methodology that is based on MC simulations and AVI that performs controlled exploration of the state space and we performed a simulation study by evaluating the behavior of the approach with respect to its tuning parameters. We also demonstrated via numerical experiments the superiority of ADP compared to traditional used rolling horizon strategies, when it comes to multistage decision problems.

Next this thesis is going to study decision making under risk sensitive criteria. We will also attempt to couple this ADP approach with the such criteria.

# CHAPTER 6

# A RISK-SENSITIVE SINGLE-PERIOD LINEAR UTILITY FOR MARKOV DECISION PROCESSES

Up to this point, we have studied risk neutral decision making in discrete event stochastic systems formulated as discounted MDPs. The objective was to maximize (minimize) the expectation of a discounted sum of stage-wise performance (cost) measure. However, when the uncertainty is significant, the spread of performance of a resulting policy may be unacceptably large to a "risk-averse" decision-maker or not wide enough for a "risk taker". Such a decision-maker may prefer to use a more flexible objective function that enables him to balance the average performance and risk according to his/her preference.

## *6.1 Introduction*

In the literature, there are essentially two ways to incorporate risk sensitivity into solving discounted MDPs. The first way is to define a single-period or stage wise utility function $U_S$, which includes a risk measure for a single time period or stage, as well as the expected reward or cost. The other is to use a multi-period utility function $U_M$, which captures a risk measure for the multi-stage performance over some time horizon. The former is the more common strategy since it is amenable to optimization strategies such as dynamic programming due to the linear nature of the objective function [48]. On the other hand, there exists no axiomatic basis to establish that this formulation will correctly account for multistage risk preferences among policies. In contrast, the latter rests on a logical formalism for time-risk tradeoffs [14, 56], but its practical usefulness may be compromised by computational difficulties in deriving the optimal policy. The readers are referred to [91] for a technical discussion on time-risk tradeoffs and to [40] for the details of axiomatic properties of different risk measures.

In this chapter, we will propose a specific form of single period utility function $U_S$

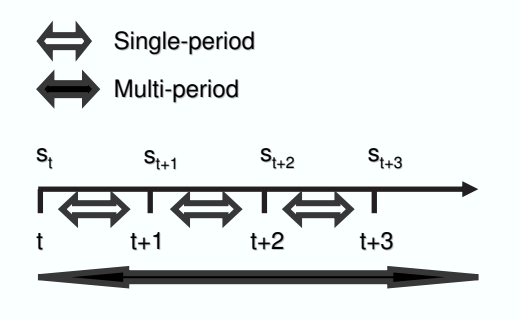



**Figure 36:** Schematic representation of: a) The Single-period utility that is applied stage wise at the reward process, b) The Multi-period utility that is applied over the summation of the stochastic reward process.

that expresses risk-time preferences. Some of the key papers appeared in 1989 and 1994, where *Filar et al* [48] addressed the problem of finding policies in discounted and undiscounted variance penalized MDPs and *Sobel* [13] in undiscounted MDPs. Apparently the risk measure used in forming their (*Filar, Sobel*) single stage utility function is the variance. Their approach is reminiscent of the earlier work of *Markowitz* [42] on portfolio analysis, where mean and variance of returns are used to formulate the problem. Specifically, the decision-maker incorporates into the objective function penalties with respect to the variability caused by a given policy. Lets denote the mean ($\mu$) and the variance ($Var$ or $\sigma^2$) associated with a single stage problem. For the single stage mean-variance optimization, there are effectively three single stage objectives that exist:

- Parametrically maximize the Langragian $\lambda\mu - (1-\lambda)\sigma^2$, as $\lambda$ spans [0,1].
- Parametrically maximize $\mu$ subject to $\sigma^2 < \lambda$ , as $\lambda$ spans $[0, \infty]$.
- Parametrically minimize $\sigma^2$ subject to $\mu > \lambda$ , as $\lambda$ spans $[0, \infty]$.

Generally, most papers explore the first objective with a fixed $\lambda$. No-one has explored the second objective and Sobel has explored the third objective via an LP formulation [13]. In principle all approaches are equivalent.

We also adopt the Lagragian approach, but instead of penalizing the variance, we will substitute the variance term with the 'conditional value at risk ($CVaR$)', which corresponds to the expected performance of the bottom tail. We conjecture that this utility function will

generate risk-sensitive policies in an intuitive manner for multistage stochastic problems modeled as discrete Markov Decision Processes (MDP's). An obvious advantage of the $CVaR$ over the variance is that it only considers the one-sided variance and not the spread of the entire distribution. Details will be given in section 6.2.

Using discounting in risk neutral MDPs is popular since these problems are more subtle and difficult without discounting. In practice one uses a discount factor close to 1 (e.g., 0.99), in order not to change the problem significantly from the original one. The usage of a discount factor along with a single period utility for risk sensitive decision-making presents a major complication factor, however. According to the discounted theorem by *Sobel* [91], discounting and single-period utility inherently instructs risk neutrality, since single period utilities are linear functions (risk neutral). We refer the reader to [91] for an in depth discussion on discounting and what it implies when it is used along with single period and multi period utility functions. Motivated by this, in this chapter we will try to establish for three simple cases the equivalence between the proposed single-stage utility function and the corresponding multi-stage utility function in terms of the resulting policy. It is revealed from our analysis that the weights in the Lagrangian function in the single-stage utility behave non-intuitively and their choice is critical in forming an effective risk-sensitive objective function.

The remainder of this chapter is organized as follows. In section 6.2, we formally present the functional form of the proposed myopic single period risk sensitive utility function. In section 6.3, we discuss its advantages over a single stage mean-variance utility and discuss briefly the exponential multi period utility, along with their corresponding optimality equations, which will be used for comparison. In section 6.4, we derive an analytical expression that establishes an equivalence between the proposed single period mean-CVaR utility and the exact multi period mean-CVaR utility for a simple case. The intuitions gained from this exercise will help us understand how the weights in the Lagrangian function should be selected for obtaining risk-sensitive policies. In section 6.5, we propose an algorithm that approximates the multistage mean-CVaR efficient frontier. In section 6.6 and 6.7, we address using this approach multi-step(multi-stage) stochastic shortest path problems with

77 and 900 discrete states, by solving the corresponding dynamic programs with both the linear and exponential utility functions and comparing the results. We also discuss convergence issues when using exponential utility functions. Despite the rich literature concerning risk neutral objectives, there is a lack of systematic numerical results of combining ADP approaches with risk-sensitive objective functions. In section 6.8, we will discuss our contribution along this direction. Finally, in section 6.9 we summarize our findings where key insights learned are highlighted.

## *6.2 The Functional Form Of The Proposed Myopic Risk Sensitive Utility*

In this section, we propose a functional form for a myopic or single period risk sensitive utility function. The risk measures that are included in this functional expression along with the expected reward are: **a)** The conditional value at risk measure, which expresses risk averseness and is denoted as $CVaR$ and **b)** The conditional value at risk taking measure, which expresses a risk taking attitude and is denoted as $CVaRT$. These risk measures will be mathematically defined for a single stage profit distribution, which is a random variable and is denoted as $z_t = f(s_{t+1}|s_t, \alpha_t)$.

The risk measure value at risk ($VaR_\eta(z_t)$) and the coherent risk measure $CVaR_\eta(z_t)$ [40] are usually meant for a loss distribution and correspond to an upper percentile of that distribution dictated by the confidence interval $\eta$. Nonetheless, since we are dealing with utilities we adjust these definitions to represent lower quantiles of a reward distribution.

The formal mathematical definitions of $VaR_\eta(z_t)$ and $CVaR_\eta(z_t)$ that correspond to such lower quantiles of a continuous profit distribution follow:

$$VaR_\eta(z_t) = min\{\zeta_1 \in \mathbb{R} : P((z_t) \leq \zeta_1) \geq \eta\} \tag{72}$$

$$CVaR_\eta(z_t) = \frac{1}{\eta} \int_{z_t \leq VaR_\eta} P(s_{t+1}|s, \alpha_t) f(s_{t+1}|s_t, \alpha_t) d\omega \tag{73}$$

$\omega$ corresponds to the realization of the uncertainty space $\Omega$ with density $P(s_{t+1}|s, \alpha_t)$.

The formal mathematical definitions of the risk taking risk measures $VaRT_\beta(z_t)$ and $CVaRT_\beta(z_t)$ hold for a given confidence interval $\beta$ and correspond to a quantile from the

upper tail of a continuous profit distribution.

$$VaRT_{\beta}(z_t) = max\{\zeta \in \mathbb{R} : (P(z_t) \geq \zeta) \leq \beta\} \tag{74}$$

$$CVaRT_{\beta}(z_t) = (1-\beta)^{-1} \int_{z_t \geq VaRT_{\beta}} P(s_{t+1}|s_t, \alpha_t) f(s_j|s_t, \alpha_t) d\omega \tag{75}$$

Schematically, the risk measures $VaR_{\eta}(z_t)$ and the $CVaR_{\eta}(z_t)$ and the corresponding Value at Risk Taking ($VaRT_{\beta}(z_t)$) and Conditional Value at Risk Taking ($CVaRT_{\beta}(z_t)$) are shown in Fig.37.

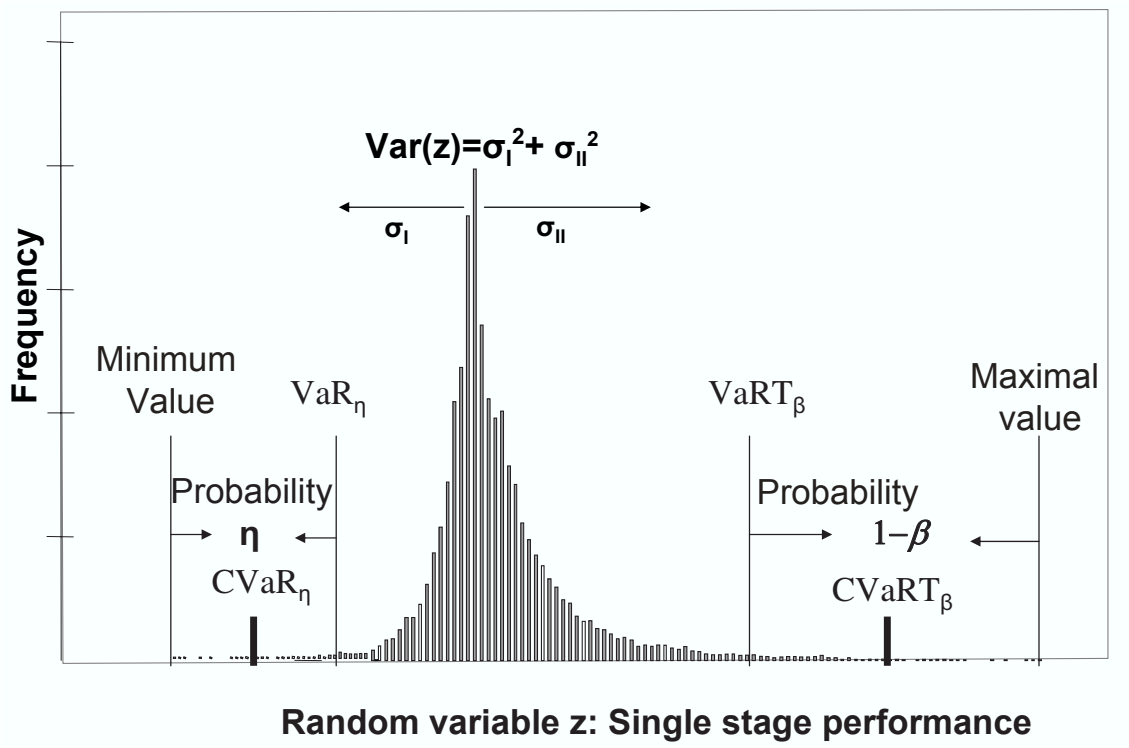



**Figure 37:** Schematic representation of the risk measures given a single stage profit distribution.

In Figure 37, by $Var(z)$ we represent the variance of the distribution with respect to the expected single stage reward, which is the summation of the corresponding semi-variances ($\sigma_I^2$ and $\sigma_{II}^2$). $\sigma_{I,t}^2$ is the semi-variance that corresponds to the density of the profit realizations smaller that the expected single stage reward for a single time period $t$, while $\sigma_{II,t}^2$ is the semi-variance that corresponds to the density of the profit realizations greater than the expected single stage reward for a single time period $t$. In the case of a normal distribution $\sigma_I = \frac{\sqrt{2}}{2}\sigma$.

Assume that the system occupies state $s_t$ and for an action $\alpha_t$ moves the system with specified conditional probability $P(s_{t+1}|s_t, \alpha_t)$ to the set of states $s_{t+1}$. The physical interpretation of the above risk measures for a single stage reward distribution is the following:

- $VaR_\eta(z_t)$: Value at risk is the minimum profit not exceeded with a given probability defined as the confidence level $\eta$, over a single period.
- $CVaR_\eta(z_t)$: This quantity represents the expectation of the lower $\eta$ percentage of the single stage profit distribution and is a representative quantity of the profit obtained by the worst cases cases with density $\eta$.
- $VaRT_\beta(z_t)$: Value at risk taking corresponds to an upper percentile of the profit distribution. This quantity represents the maximum profit not exceeded with a given probability defined as the confidence level $1-\beta$, over a single time period.
- $CVaRT_\beta(z_t)$ is the expectation of the 'optimistic' tail of the profit distribution with total density $1-\beta$. Intuitively the coherent measure $CVaRT_\beta$ corresponds to a very optimistic estimation of the single profit under a specific action $\alpha_t$.

The functional form of the proposed single stage risk sensitive linear utility for discrete event systems is:

$$U_S(s_t, \alpha_t, \lambda) = \begin{cases} \lambda \sum_{s_{t+1}} P(s_{t+1}|s_t, \alpha_t) f(s_{t+1}|s_t, \alpha_t) + (1-\lambda) CVaR_\eta(z_t) & \text{,if Risk Averse} \\ \sum P(s_{t+1}|s_t, \alpha_t) f(s_{t+1}|s_t, \alpha_t) & \text{,if Risk Neutral} \\ \lambda \sum P(s_{t+1}|s_t, \alpha_t) f(s_{t+1}|s_t, \alpha_t) + (1-\lambda) CVaRT_\beta(z_t) & \text{,if Risk Taker} \end{cases}$$

This proposed myopic utility, if used in a DP framework is expected to be able to represent both risk averse and risk sensitive multi-stage attitudes. In this chapter, we will focus on representing risk averse attitudes in a specific class of problems often called Goal Directed Markov Decision Processes (GDMDP).

The specific characteristic of those problems is that they have a dedicated set of goal states , denoted as $\mathbb{G}$. All these states within this set are self absorbing meaning that you cannot escape from them under any action and their optimal value function is 0. For the numerical applications, we will use the single stage mean-$CVaR_\eta$ functional form within

the system of optimality equations for general multistage instances with a specified set of goal states and we will assume all the rewards to be non-positive quantities.

$$
\begin{aligned}
&J^{\pi}(s_t,\lambda,\gamma) = \max_{\alpha_t \in \mathbb{A}} U_S(s_t,\alpha_t,\lambda) + \gamma \sum_{s_{t+1}\in\mathbb{S}} Pr(s_{t+1}|s_t,\alpha_t) J^{\pi}(s_{t+1},\lambda), && s_t \in \mathbb{S} \\
&J^{\pi}(s_t,\lambda,\gamma) = 0, && s_t \in \mathbb{S} \setminus \mathbb{G}
\end{aligned}
\tag{76}
$$

The main advantage of using this functional form as a single stage utilities, along with the discount factor, is that one can use the traditional optimality equations in exact DP and ADP procedures. This fact is mainly attributed to the nice properties of the expectation operator with respect to linear utilities and to the fact that the usage of discount factor guarantees the convergence of DP, under mild assumptions. These assumptions are mentioned at the second chapter.

A second important advantage of this proposed single period utility, compared to other commonly used single period utilities(e.g.mean-variance), is that it can handle efficiently heavily skewed distributions. Moreover, for general multistage optimization problems the single stage mean-$CVaR_{\eta}$ calculation can be achieved via the LP formulation as proposed by *Rockafellar and Uryasef* [50]. Given this fact one can address the COD with respect to the evaluation of the $CVaR_{\eta}$ or $CVaRT_{\beta}$ by creating a large scale LP via sampling.

## *6.3 Single Stage Mean-$CVaR_{\eta}$ Vs Single Stage Mean-Variance And Exponential Multistage Utility*

In the numerical instances in section 6.6 the summation of the proposed utility is evaluated along with the summation of the mean-variance single period utility and an exponential multistage utility. In this section, for conceptual as well as for practical comparison purposes, we analyze in a qualitative manner the difference between optimizing the summation of the single stage mean-CVaR utility versus the summation of the mean-variance single period utility in a multistage setting with normally distributed rewards. Therefore, we create general shortest path examples where the summation of the proposed single period utility will yield a wider spectrum of policies than the summation of the single period mean-variance utility. Moreover, we delineate the usage of the exponential utilities as multi-stage

utilities within MDPs along with their corresponding optimality equations.

### 6.3.1 Qualitative Difference On Optimizing The Summation Of Single Period Mean-CVaR$_\eta$ And Mean-Variance Utilities

To simplify the notation assume that for a state-action pair $(s_t, \alpha_t)$: a) $\hat{x}_t$ is the expected single stage reward, b) $\sigma^2_{I,t}$ is the semi-variance that corresponds to the density of profit realizations smaller that $\hat{x}_t$ for a single time period $t$, c) $\sigma^2_{II,t}$ is the semi-variance that corresponds to the density of profit realizations greater that $\hat{x}_t$ for a single time period $t$ [1] d) the confidence interval is set to $\eta = 0.05$ and $\beta = 0.95$. To calculate the one step $CVaR_\eta$ for normal distributions, we use $CVaR_\eta = \hat{x}_t$ - $g(\eta)\sigma_I$, where $g(\eta) = \sqrt{2}\exp(erf(2\eta - 1)^2)^{-1}(1-\eta)^{-1}$.$(g(0.05) = 0.1086)$

Using the proposed mean-$CVaR_\eta$ single stage utility and by setting the Langragean parameter $\lambda = \lambda_1$ , we try to find the sequence of actions $\{\alpha_0, \alpha_1, ..., \alpha_{t-1}, \alpha_t\}$ that will maximize the following expression.

$$\begin{aligned}\max_{\alpha_t} \sum_t \gamma^t(\lambda_1 \hat{x}_t + (1-\lambda_1)CVaR_\eta(z)) &= \max_{\alpha_t}(\sum_t \gamma^t(\lambda_1 \hat{x}_t + (1-\lambda_1)(\hat{x}_t - g(\eta)\sigma_{I,t}))) \\ &= \frac{1}{1-\gamma}\max_{\alpha_t}(\sum_t (\hat{x}_t - (1-\lambda_1)g(\eta)\sigma_{I,t}))\end{aligned}$$

- When $\lambda_1 = [0, 1)$ we effectively maximize two objectives: The first is the maximization of the expectation of a discounted sum of stage-wise rewards, while the second objective is the minimization of the expectation of a discounted sum of the single stage semi-variances $\sigma^2_{I,t}$

  Depending on the choice of $\lambda_1$ and $\eta$, we tune the single stage ratio of the weights between these two objectives.

- For $\lambda_1 = 1$, the single stage mean-CVaR utility is risk neutral and the proposed formulation maximizes the expectation of a discounted sum of stage-wise rewards.

When using the single stage mean-variance utility to solve the multi-stage optimization problem, we adopt the following optimization problem. (Here, the Langragean parameter

[1] The statistics $\sigma^2_I$ and $\sigma^2_{II}$ are illustrated in Fig.37.

$\lambda = \lambda_2$):

$$\max_{\alpha_t} \sum_t \gamma^t(\lambda_2 \hat{x}_t - (1-\lambda_2)Var(z)) \quad = \quad \frac{1}{1-\gamma} \max_{\alpha_t}(\sum_t (\lambda_2 \hat{x}_t - (1-\lambda_2)(\sigma^2_{I,t} + \sigma^2_{II,t})))$$

- When $\lambda_2 = 1$ this utility corresponds to a risk neutral solution.
- When $\lambda_2 = (0, 1)$ we consider three objectives: The first is the maximization of the expectation of a discounted sum of stage-wise rewards with single stage weight $\lambda_2$. The other two objectives are the minimization of the expectation of a discounted sum of the single stage semi-variances $\sigma^2_{I,t}$ and $\sigma^2_{II,t}$ respectively. These later objectives are weighted at each stage with respect to a parameterized weight (1-$\lambda_2$),(1-$\lambda_2$). This objective obviously tries to minimize the standard deviation with respect to deviations that are greater than $\hat{x}_t$. From a decision making perspective this formulation is uncomfortable and does not have an intuitive basis, since no-one would include as penalties both such deviations, while trying to maximize the expected profit.
- When $\lambda_2 = 0$ we minimize the expectation of a discounted sum of the single stage semi-variances. Again, this is an uncomfortable formulation for the same reasons as discussed above.

What follows demonstrates two general shortest path settings for which the proposed summation of the mean-CVaR utility has the capacity to yield a wider spectrum of policies than the summation of the single period mean variance utility.

### 6.3.2 Multi Step Shortest Path Examples

These general multi step shortest path examples with a single starting and goal state are illustrated in Figure 38. Here, we assume that the uncertainty affects the single stage reward stream and not at transition probabilities. The first shortest path example demonstrates the capacity of the proposed utility to retrieve risk seeking policies, while the second shortest path example demonstrated the fact that it can efficiently handle problems composed out of single stage shewed reward distributions. In these problems, we consider performance maximization with each single stage profit to be a **non positive quantity** and the goal state to be an absorbing state.

The data for the first example appear in Figure 38:

- If we use the summation of the single-period mean variance utility and maximize the expected performance we would follow a route composed out of the rectangles. Moreover if we minimize the variance of a policy, we would follow the same route since the multistage variance of a policy composed out of deterministic costs is 0. Therefore, this objective can only yield a specific policy since mean and variance are not contradicting.

- If we use the summation of the mean-$CVaRT_\beta$ single stage risk seeking utility and maximize the expected mean we would follow the route composed out of rectangles. But, if we adopt a risk taking attitude, we would find solutions with respect to the mean-$CVaRT_\beta$ risk measure. If this single stage performance-expression: $\lambda_1 \hat{x}_t + (1 - \lambda_1) CVaRT_\beta(z) > -A$ holds for a combination of values for the parameters $\lambda_1, \beta$ then this utility will instruct a solution-route composed out of the circle states.

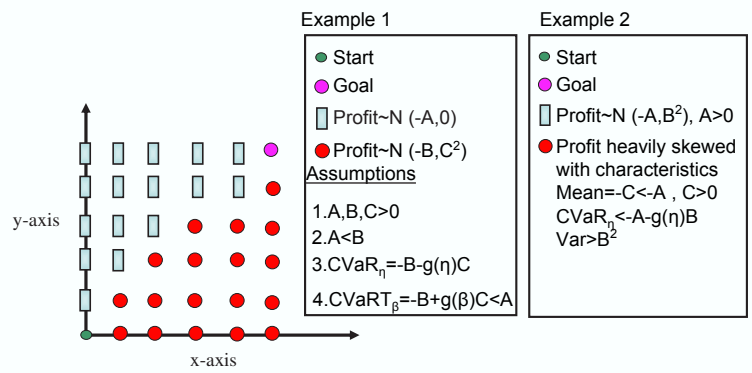



**Figure 38:** Examples for which our proposed objective yields a wider spectrum of policies than the mean variance formulation.

The data for the second example appear in Figure 38:

- If we use the summation of the mean variance as the objective function and maximize the mean we would follow the route composed out of rectangles. Moreover, if we minimize the variance of a policy, we would follow the same route, since the single stage variance of the rectangles $B^2$ is assumed to be less than the single stage variance of the skewed distribution of the circle states. Therefore, this objective can only yield

a specific policy since mean and variance are not contradicting. This route would be composed out of rectangle states.

- If we use the summation of the mean-$CVaR_\eta$ risk averse utility as the objective and maximize the mean we would follow the route composed out of rectangles. But, if we wanted to express a risk averse attitude by using this Langragean utility, that would eliminate the worst cases and maximize the systems performance, then under the condition that the single stage $-C - CVaR_\eta$ of the skewed single stage profit distribution is more than $-A - g(\eta)B$, we would be able to produce a policy going through a route composed out of circle states.

In summary, the proposed utility can yield a wider spectrum of policies than the mean variance formulation.

### 6.3.3 Multistage Exponential Utility Function

*Corner and Corner* [92] explicitly refers that in almost 30% of their reviewed applications people adopt exponential utility functions, while in two thirds of them they use risk neutral formulations (linear utilities). More complex non linear utility functions (e.g. in the form of one switch utilities) have not been used in recent Operations Research (OR) literature, since, if such utility is used, the resulting multistage optimization is a complex non decomposable problem.

For non linear multi stage utilities *Liu and Koenig* [56] make the problem decomposable by augmenting the state space with an additional state variable. By doing that, one can utilize valid optimality equations but then will violate one of the basic assumptions (finite $\mathbb{S}$) that guarantees the convergence of the DP operator to the optimal value function. Therefore the convergence properties of DP are effectively destroyed.

To axiomatically marry discounting and risk sensitivity (time-risk preference), we need to introduce a formalism which invites an exponential multi stage utility function [56]. In fact this is the only meaningful choice for multi period utilities, since the inverse of a single stage exponential utility is a logarithmic function and represents the single stage Certainty Equivalence (Section 2.6). This logarithmic function can be additive over multi stages and

because of that it yields a form of recursive DP equations as shown explicitly in (*Howard and Matheson* [93].

In this chapter, the practical implementation of the exponential utility function will be used as a measure of comparison for the proposed single stage utility. An exponential form of multi-period utilities is:

$$U_M(W) = \begin{cases} \xi^W, & \xi > 1 \text{ (Risk Takers)} \\ -\xi^W, & 0 < \xi < 1 \text{ (Risk Averse)} \end{cases}$$

$W$ is the wealth level corresponding to a reward stream produced by a given policy over a given horizon.

An excellent treatment concerning the properties of these utilities, a complete literature review, as well as their application to realistic applications is given in Liu's [14] thesis.

What follows is a brief historic review of how the multi-period exponential functions were used within finite and infinite MDP's. The finite horizon problem using multistage exponential utilities was first formulated in (*Howard and Matheson* [93]). They studied how to find optimal policies in the class of Markovian Determinisitic policies for finite horizon problems, and obtained the system of optimality equations, which have a unique solution. Mainly, they used the property that the inverse of a single stage exponential function, which is a logarithic function is additive over the stages. Based on the derived optimality equations, they showed that the finite horizon problems can be solved using a backward induction procedure.

One of the first that maximized the multistage expected exponential utility of total discounted rewards over an infinite horizon was *Jaquette* [94]. Similarly, *Chung and Sobel* [12] studied this objective for risk-averse agents and finite models with nonnegative rewards and obtained a system of optimality equations. The derived system of equations also appears in *Coraluppi and Marcus Avila-Godoy* [47] proved the correctness of the system of these optimality equations for multistage exponential utilities for all types policies (randomized+deterministic). Those equations will be used along this chapter.

#### 6.3.3.1 Optimality Equations Of The Multi period Exponential Utility Functions And Properties

The optimality equations for GDMDP's with non positive rewards differ for exponential utilities as shown in *Avila-Godoy* [47] are displayed below:

$$\begin{aligned} &J_M^{i+1}(s_t,\xi) = \max_{\alpha_t \in \mathbb{A}} \sum_{s_{t+1} \in \mathbb{S}} Pr(s_{t+1}|s_t,\alpha_t)\xi^{f(s_{t+1}|s_t,\alpha_t)} J_{Inter}^{i}(s_{t+1}), && s \in \mathbb{S} \setminus \mathbb{G} \\ &J_M(s_t,\xi) = signln\xi, && s \in \mathbb{G} \end{aligned} \tag{77}$$

, where the sign function is a mathematical function that extracts the sign of a real number. It is defined as follows:

$$sign(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}$$

These optimality equations coincide for discounted and undiscounted problems. The term $\sum_{s_{t+1} \in \mathbb{S}} Pr(s_{t+1}|s_t,\alpha_t)\xi^{f(s_{t+1}|s_t,\alpha_t)}$ represents the so called pseudo probabilities since they do not necessarily sum to one. More details about the interpretation of the pseudo-probabilities in a discounted or undiscounted setting can be found in *Liu* [14].

This section summarizes important properties about the class of exponential functions:

- Why does $\xi < (>)1$ signify risk averse (seeking) behavior ? Let $i = signln\xi$, then $U_M(f) = i\xi^w$. The risk measure [14] for this function is:

  $$R_M(W) = \frac{U_M^{''}(W)}{U_M^{'}(W)} = -log\xi$$

  When $\xi > 1$, the utility function is convex and $R_M(W) < 0$, indicating the agent is risk-seeking. While when $0 < \xi < 1$, the utility function is concave and $R_M(W) > 0$, indicating the agent is risk-averse.

  Varying the parameter $\xi$ and solving the DP recursion, we will be able to solve for different decision maker attitudes towards risk.

- The main problem with $U_M$ is that the value function may or may not exist. In practise, the conditions needed to ensure optimality are finite state and action space. Moreover, the value iteration converges to the optimal values starting from an initial value function where for all states $i \geq J_M^0(s_t) \geq J_M^*(s_t)$ *Avila-Godoy.* [47].

## *6.4 The Source Of Deviation Between The Summation Of The Single Stage Mean-CVaR Utility Vs The Exact Multi-stage Mean-CVaR Utility*

The purpose of this section is to evaluate the summation of the single stage linear mean-CVaR utility against the exact multi-period mean-CVaR utility on two single stage but multi-step problems

This evaluation will induce useful intuition about the weighting factors on the statistics involved to the optimization problem.

### 6.4.1 Problem Statement 1

The problem statement is the following. Assume that the system can move only along a deterministic trajectory and for each time $t$ the system is realizing a discounted reward from the same normal distribution $\gamma^t f_t \sim N(\gamma^t \mu, (\gamma^t \sigma)^2)$.

What follows is the analytic evaluation of the proposed single period mean-CVaR utility summed over an infinite horizon. The following equalities: $\sum_{t=0}^{\infty} \gamma^t = \frac{1}{1-\gamma}$ and $\sum_{t=0}^{\infty} (\gamma^t)^2 = \frac{1}{1-\gamma^2}$, where $\gamma < 1$ are used in deriving the following results.

$$\begin{aligned} \sum_{t=0}^{\infty} \gamma^t (\lambda_1 \mathbb{E}(f_t) + (1-\lambda_1) CVaR_\eta(f_t)) &= \sum_{t=0}^{\infty} \gamma^t (\lambda_1 \mu + (1-\lambda_1)(\mu - g(\eta)\sigma_I) \\ &= \frac{1}{1-\gamma}(\lambda_1 \mu + (1-\lambda_1)(\mu - g(\eta)\sigma_I) \\ &= \frac{1}{1-\gamma}\mu - \frac{(1-\lambda_1)}{1-\gamma} g(\eta)\sigma_I \end{aligned} \tag{78}$$

What follows next is the analytic evaluation of the exact multi period mean-CVaR utility on the same problem over an infinite horizon. To perform this evaluation, we note that the

infinite discounted summation of the functions $f_t$ result to the following normal multistage distribution $D$.

$$D = \sum_{t=0}^{\infty} \gamma^t f_t \sim N\left(\frac{1}{1-\gamma}\mu, \frac{\sigma^2}{1-\gamma^2}\right)$$

Therefore the multi-stage mean-$CVaR_\eta$ tradeoff becomes:

$$\begin{aligned} \lambda^*\mathbb{E}(D) + (1-\lambda^*)CVaR_\eta(D) &= \lambda^*\frac{1}{1-\gamma}\mu - (1-\lambda^*)\left(\frac{1}{1-\gamma}\mu + g(\eta)(\sqrt{\frac{1}{1-\gamma^2}}\sigma_I\right) \\ &= \frac{1}{1-\gamma}\mu - (1-\lambda^*)g(\eta)\sqrt{\frac{1}{1-\gamma^2}}\sigma_I \end{aligned} \tag{79}$$

For this simple problem, these two objectives evaluate the same policy in the following manner: they put an equal weight $\frac{1}{1-\gamma}$ on the expected mean, but the main difference lies in the weight assigned to the standard deviation of the semi-variance $\sigma_I^2$.

- The summation of the single period discounted mean-$CVaR_\eta$ assigns the functional weight $W_1(\lambda_1, \gamma, \eta) = \frac{1-\lambda_1}{1-\gamma}g(\eta)$ to the standard deviation of the semi-variance $\sigma_I^2$ : .
- The multi period discounted mean-CVaR formulation assigns the following functional form of weight $W_2(\lambda^*, \gamma, \eta)$ to the standard deviation of the semi-variance $\sigma_I^2$ : $W_2(\lambda^*, \gamma, \eta) = (1-\lambda^*)g(\eta)\sqrt{\frac{1}{1-\gamma^2}}$.
- Those two approaches would result into the same this policy if and only if $W_1(\lambda_1, \gamma, \eta) = W_2(\lambda^*, \gamma, \eta)$. From this, we can derive the following equivalence relationship:

  $$\lambda_1 = 1 - (1-\lambda^*)(1-\gamma)\sqrt{\frac{1}{1-\gamma^2}}$$

  If the problem is multi stage ($0 < \gamma < 1$) and not myopic ($\gamma = 0$) then:

  - $\lambda^* = 0 \Longrightarrow \lambda_1 = 1 - (1-\gamma)\sqrt{\frac{1}{1-\gamma^2}}$.

    For $\gamma = 0.1$ and $\lambda^* = 0$ then $\lambda_1 = 0.0955$

    For $\gamma = 0.5$ and $\lambda^* = 0$ then $\lambda_1 = 0.4226$

    For $\gamma = 0.99$ and $\lambda^* = 0$ then $\lambda_1 = 0.9291$

    For $\gamma = 0.999$ and $\lambda^* = 0$ then $\lambda_1 = 0.9776$.

  - $\lambda^* = 1 \Longrightarrow \lambda_1 = 1$

- $0 < \lambda^* < 1 \Longrightarrow 1 - (1-\gamma)\sqrt{\frac{1}{1-\gamma^2}} < \lambda_1 < 1$

  Therefore, for this simple problem, for any $(\gamma, \lambda^*)$, there is a $\lambda_1$ used at the summation of the single stage mean-$CVaR_\eta$ that evaluates the same as the multi stage mean - $CVaR_\eta$.

Otherwise, if the problem is myopic $\lambda^* = \lambda_1$.

- The corresponding surfaces to the weight functions $W_1(\lambda_1, \gamma, \eta)$ and $W_2(\lambda^*, \gamma, \eta)$ are illustrated in Figure 39. We observe that, when the range of the risk averse parameters $\lambda_1$,$\lambda^*$ is closer to one and also the discount factor is greater than 0.8, then the summation of the single period discounted mean-CVaR utility functions results in a better approximation of the weight prescribed by the multi period discounted mean-CVaR utility than if $\lambda_1$,$\lambda^*$ is closer to zero. In fact when the risk averse $\lambda_1$,$\lambda^*$ is closer to 0, we notice a large discrepancy between the two surfaces that prescribe the corresponding weights. In this case for this simple problem, we realize that our proposed utility (Section 6.2) assigns a severe weight on minimizing the $\sigma_{A,I}$. The derived policy is not expected to correspond to a meaningful solution with respect to the multi period mean-$CVaR$ efficient frontier. If fact if $\lambda_1$ is set close to 0 and $\gamma > 0.8$, given our previous discussion there is no hope that the proposed objective will evaluate the same as one solution belonging to the efficient frontier of the exact multi stage mean-$CVaR$.

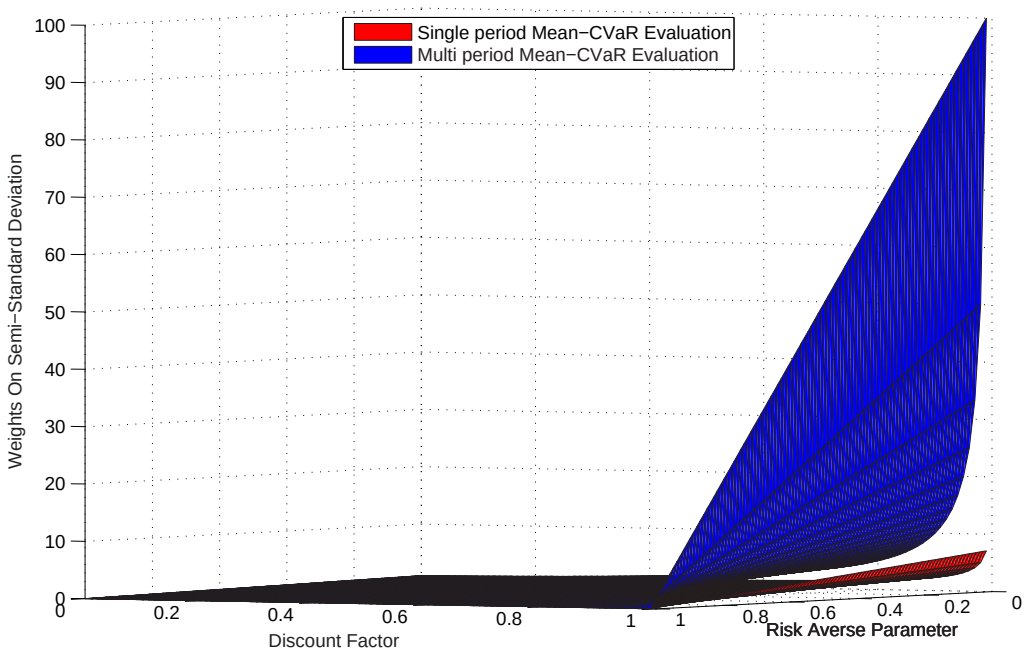



**Figure 39:** We demonstrate the difference in the weighting of the semi-standard deviation, when evaluating the discounted multistage mean-CVaR trade-off on the entire distribution against the summation of the discounted single period utility (Eq.78-79).

### 6.4.2 Problem Statement 2

First, assume that the system can move only along a deterministic trajectory. For the first $\nu_A$ time periods the system realizes a discounted reward from the following normal distribution $\gamma^t g_{A,t} \sim N(\gamma^t \mu_A, (\gamma^t \sigma_A)^2)$, then for the following $\nu_B$ time periods the system realizes an discounted reward from the following normal distribution $\gamma^t g_{B,t} \sim N(\gamma^t \mu_B, (\gamma^t \sigma_B)^2)$.

The evaluation of the summation of the single stage mean-CVa$R_\eta$ for a specific choice of $\eta$ when applied to this problem follows:

$$\begin{aligned}\sum_{t=0}^{\nu_A-1} \gamma^t(\lambda_1 \mathbb{E}(g_{A,t}) + (1-\lambda_1) CVaR_\eta(g_{A,t})) &= \sum_{t=0}^{\nu_A-1} \gamma^t(\lambda_1 \mu_A + (1-\lambda_1)(\mu_A - g(\eta)\sigma_{A,I}) \\ &= \sum_{t=0}^{\nu_A-1} \gamma^t(\lambda_1 \mu_A + (1-\lambda_1)(\mu_A - g(\eta)\sigma_{A,I}) \\ &= \big(\sum_{t=0}^{\nu_A-1} \gamma^t\big)\mu_A - \big(\sum_{t=0}^{\nu_A-1} \gamma^t\big)(1-\lambda_1) g(\eta)\sigma_{A,I}\end{aligned} \tag{80}$$

$$\begin{aligned}\sum_{t=\nu_A}^{\nu_A+\nu_B} \gamma^t(\lambda_1 \mathbb{E}(g_{B,t}) + (1-\lambda_1) CVaR_\eta(g_{B,t})) &= \sum_{t=\nu_A}^{\nu_A+\nu_B} \gamma^t \nu_B(\lambda_1 \mu_B + (1-\lambda_1)(\mu_B - g(\eta)\sigma_{B,I}) \\ &= \sum_{t=\nu_A}^{\nu_A+\nu_B} \gamma^t(\lambda_1 \mu_B + (1-\lambda_1)(\mu_B - g(\eta)\sigma_{B,I}) \\ &= \big(\sum_{t=\nu_A}^{\nu_A+\nu_B} \gamma^t\big)\mu_B - \big(\sum_{t=\nu_A}^{\nu_A+\nu_B} \gamma^t\big)(1-\lambda_1) g(\eta)\sigma_{B,I}\end{aligned} \tag{81}$$

$$\begin{aligned}&\sum_{t=0}^{\nu_A-1} \gamma^t(\lambda_1 \mathbb{E}(g_{A,t}) + (1-\lambda_1) CVaR_\eta(g_{A,t})) + \sum_{t=\nu_A}^{\nu_A+\nu_B} \gamma^t(\lambda_1 \mathbb{E}(g_{B,t}) + (1-\lambda_1) CVaR_\eta(g_{B,t})) \\ &= \big(\sum_{t=0}^{\nu_A-1} \gamma^t\big)\mu_A - \big(\sum_{t=0}^{\nu_A-1} \gamma^t\big)(1-\lambda_1) g(\eta)\sigma_{A,I} + \big(\sum_{t=\nu_A}^{\nu_A+\nu_B} \gamma^t\big)\mu_B - \big(\sum_{t=\nu_A}^{\nu_A+\nu_B} \gamma^t\big)(1-\lambda_1) g(\eta)\sigma_{B,I} \\ &= \big(\sum_{t=0}^{\nu_A-1} \gamma^t\big)\mu_A + \big(\sum_{t=\nu_A}^{\nu_A+\nu_B} \gamma^t\big)\mu_B - (1-\lambda_1) g(\eta)\big(\big(\sum_{t=0}^{\nu_A-1} \gamma^t\big)\sigma_{A,I} + \big(\sum_{t=\nu_A}^{\nu_A+\nu_B} \gamma^t\big)\sigma_{B,I}\big)\end{aligned} \tag{82}$$

Next, we will evaluate the multi stage mean - CVa$R_\eta$ distribution D and will compare it

against Eq.(82).

$$\begin{aligned} D &= \gamma^0 g_{A,0} + \gamma^1 g_{A,1} + ... + \gamma^{\nu_A} g_{A,\nu_A} + ... + \gamma^{\nu_A+1} g_{B,\nu_A+1} + ... + \gamma^{\nu_A+\nu_B} g_{B,\nu_A+\nu_B} \\ &= (\big(\sum_{t=0}^{\nu_A-1} \gamma^t\big)\mu_A + \big(\sum_{t=\nu_A}^{\nu_A+\nu_B} \gamma^t\big)\mu_B, (((\big(\sum_{t=0}^{\nu_A-1} \gamma^t\big)\sigma_A)^2 + (\big(\sum_{t=\nu_A}^{\nu_A+\nu_B} \gamma^t\big)\sigma_B)^2) \end{aligned} \tag{83}$$

If we set $M_1 = \big(\sum_{t=0}^{\nu_A-1} \gamma^t\big)\mu_A + \big(\sum_{t=\nu_A}^{\nu_A+\nu_B} \gamma^t\big)\mu_B$ and
$M_2 = (\big(\sum_{t=0}^{\nu_A-1} \gamma^t\big)\sigma_{A,I})^2 + (\big(\sum_{t=\nu_A}^{\nu_A+\nu_B} \gamma^t\big)\sigma_{B,I})^2$, then the multi-stage evaluation for a risk averse parameter $\lambda^*$ and a confidence interval $\eta$ becomes:

$$\begin{aligned} \lambda^* \mathbb{E}(D) + (1-\lambda^*) CVaR_\eta(D) &= \lambda^* M_1 + (1-\lambda^*)(M_1 - g(\eta)\sqrt{M_2}) \\ &= M_1 - (1-\lambda^*) g(\eta)\sqrt{M_2} \end{aligned} \tag{84}$$

One can observe that the two evaluated approaches put an equal weight on the expected mean. The main difference lies in the weights assigned to the standard deviation of the semi-variances $\sigma_{A,I}$ and $\sigma_{B,I}$.

From Eq.(82) and Eq.(84), we can derive an analytic expression for which
$\lambda_1 = f(\lambda^*, \gamma, \sigma_{I,A}, \sigma_{I,B})$. The expression follows:

$$\lambda_1 = 1 - (1-\lambda^*) \frac{\sqrt{(\big(\sum_{t=0}^{\nu_A-1} \gamma^t\big)\sigma_{A,I})^2 + (\big(\sum_{t=\nu_A}^{\nu_A+\nu_B} \gamma^t\big)\sigma_{B,I})^2}}{\big(\sum_{t=0}^{\nu_A-1} \gamma^t\big)\sigma_{A,I} + \big(\sum_{t=\nu_A}^{\nu_A+\nu_B} \gamma^t\big)\sigma_{B,I}} \tag{85}$$

In the limiting case, where the system realizes rewards for $\nu_A = k_1$ time periods and $\sum_{t=0}^{k_1} \gamma^t = \frac{1}{1-\gamma}$ and $\sum_{t=0}^{k_1} (\gamma^t)^2 = \frac{1}{1-\gamma^2}$, then Eq.(85) is the same with the result from the previous section:

$$\begin{aligned} \lambda_1 &= 1 - (1-\lambda^*) \frac{\sqrt{\frac{1}{1-\gamma^2}\sigma_{A,I}^2}}{\frac{1}{1-\gamma}\sigma_{A,I}} \\ \lambda_1 &= 1 - (1-\lambda^*)(1-\gamma)\sqrt{\frac{1}{1-\gamma^2}} \end{aligned} \tag{86}$$

If we set $\lambda^*$=0 at Eq.(85) then:

$$\lambda_1 = 1 - \frac{\sqrt{((\sum_{t=0}^{\nu_A-1}\gamma^t)\sigma_{A,I})^2 + ((\sum_{t=\nu_A}^{\nu_A+\nu_B}\gamma^t)\sigma_{B,I})^2}}{(\sum_{t=0}^{\nu_A-1}\gamma^t)\sigma_{A,I} + (\sum_{t=\nu_A}^{\nu_A+\nu_B}\gamma^t)\sigma_{B,I}} \tag{87}$$

Let's denote with $\nu'_A = \sum_{t=0}^{\nu_A}\gamma^t$ and $\nu'_B = \sum_{t=\nu_A}^{\nu_A+\nu_B}\gamma^t$. Assuming that the following statement is true, if we end up in a true statement then our initial assumption will be true:

$$\begin{aligned}
\frac{\sqrt{\frac{(\nu'_A\sigma_{A,I})^2}{2} + \frac{(\nu'_B\sigma_{B,I})^2}{2}}}{\nu'_A\sigma_{A,I} + \nu'_B\sigma_{B,I}} &< 1 \\
\sqrt{\frac{(\nu'_A\sigma_{A,I})^2}{2} + \frac{(\nu'_B\sigma_{B,I})^2}{2}}) &< \nu'_A\sigma_{A,I} + \nu'_B\sigma_{B,I} \\
\frac{(\nu'_A\sigma_{A,I})^2}{2} + \frac{(\nu'_B\sigma_{B,I})^2}{2}) &< (\nu'_A\sigma_{A,I} + \nu'_B\sigma_{B,I})^2 \\
\frac{(\nu'_A\sigma_{A,I})^2}{2} + \frac{(\nu'_B\sigma_{B,I})^2}{2}) &< (\nu'_A\sigma_{A,I})^2 + (\nu'_B\sigma_{B,I})^2 + 2\nu'_A\nu'_B\sigma_{A,I}\sigma_{B,I} \\
0 &< (\frac{(\nu'_A\sigma_{A,I})^2}{2} + \frac{(\nu'_B\sigma_{B,I})^2}{2}) + 2\nu'_A\nu'_B\sigma_{A,I}\sigma_{B,I}
\end{aligned}$$

The last statement is true since $\nu'_A, \nu'_B, \sigma_{A,I}, \sigma_{B,I} > 0$. That means that for this simple problem, if $\lambda^* = 0$ then the summation of these linear mean-CVa$R_\eta$ utilities will yield the same evaluation as the multi stage mean $CVaR_\eta$ if and only if $0 < \lambda_1 < 1$.

As explained before to quantify, how close to 1 should the $\lambda_1$ parameter be in order to yield the same evaluation with $\lambda^* = 0$ is a problem specific task. Trivially, $\lambda^* = \lambda_1 = 1$ also yields the same evaluation, which is a risk neutral evaluation. By performing this analysis one can identify the range of $\lambda_1$ that one can use linear mean-$CVaR$ functions and still produce meaningful policies that belong to the multi stage mean-$CVaR$ efficient frontier. Without loss of generality such analysis can be carried over to the undiscounted version of this problem.

Based on these intuitive conclusions, we will propose a general procedure that utilizes linear intra-period mean-CVaR utilities and targets to approximate the multi-stage mean-CVaR efficient frontier.

### 6.4.3 Problem Statement 3

This problem statement evaluates the two pre-mentioned objectives on a time correlated stage-wise profit function, which is expressed via: $f(t+1) = \psi f(t) + (1-\psi)w(t)$, where $w(t)$ is an i.i.d. sequence. The assumptions needed to derive similar analytical results as in problem statements 1,2 are: 1) $f(0) \sim N(\mu, \sigma^2)$ and 2) $w(t) \sim N(\mu, \sigma^2)$.

The main difference between this problem statement and the ones appeared previously is the time correlation aspect. At time period $t = 0$, we will be drawing a sample from the normal distribution $f(0)$, which is going to propagate given the dynamic equation $f(t+1) = \psi f(t) + (1-\psi)w(t)$.

If $\psi = 1$ that means that the sample drawn at time $t = 0$ will be propagated via the dynamic equation until the end of the horizon which is $k$ steps $f(0) = f(1) = f(2) = f(3) =$ ... . For this case the exact multi-stage mean-$CVaR$ objective and its proposed surrogate objective coincide.

If $\psi = 0$ the problem statement 3 coincides with the problem statement 1. To facilitate the discussion we analytically derive the distributions $\gamma^0 f(0), \gamma^1 f(1), \gamma^2 f(2), ....$

$$
\begin{aligned}
\gamma^0 f(0) &\sim N\big(\mu, \sigma^2\big) \\
\gamma^1 f(1) &= \gamma\big(\psi f(0) + (1-\psi)w(1)\big) \\
&\sim N\big((\psi + (1-\psi))\gamma\mu, (\psi^2 + (1-\psi)^2)(\gamma\sigma)^2\big) \\
\gamma^2 f(2) &= \gamma^2\big(\psi^2 f(0) + \psi(1-\psi)w(1) + (1-\psi)w(2)\big) \\
&\sim N\big((\psi^2 + (1-\psi)(1+\psi))\gamma^2\mu, (\psi^4 + (1-\psi)^2(1+\psi^2)(\gamma^2\sigma)^2\big) \\
&\ . \ . \ . \\
\gamma^k f(k) &= \gamma^k\big(\psi^k f(k) + \psi^{k-1}(1-\psi)w(1) + \psi^{k-2}(1-\psi)w(2) + ... + (1-\psi)w(k)\big) \\
&\sim N\big((\psi^k + (1-\psi)(1+\psi+\psi^2+...+\psi^{k-1}))\gamma^k\mu, \\
&\qquad (\psi^{2k} + (1-\psi)^2(1+\psi^2+\psi^4+...+\psi^{2k)})(\gamma^k\sigma)^2\big) \\
&\ . \ . \ . \\
&\ . \ . \ .
\end{aligned}
$$

What follows is the analytic evaluation of the exact multi period mean-CVaR utility for $\infty$ time steps. To perform this evaluation, we sum $\infty$ discounted normal distributions that will consist the multistage distribution $D = \sum_{t=0}^{\infty} \gamma^t f(t)$.

$$
\begin{aligned}
D &= \gamma^0 f(0) + \gamma^1 f(1) + \gamma^2 f(2) + ... + \gamma^k f(k) \\
&= f(0) + \gamma(\psi f(0) + (1-\psi)w(1)) + \gamma^2(\psi^2 f(0) + \psi(1-\psi)w(1) + (1-\psi)w(2)) \\
&\quad +... + \gamma^k(\psi^k f(k) + \psi^{k-1}(1-\psi)w(1) + \psi^{k-2}(1-\psi)w(2) + ... + (1-\psi)w(k)) + ... \\
&= \frac{1}{1-\gamma\psi} f(0) + \left(\frac{\gamma}{1-\gamma\psi} - \frac{\gamma\psi}{1-\gamma\psi}\right) w(1) + \left(\frac{\gamma^2}{1-\gamma\psi} - \frac{\gamma^2\psi}{1-\gamma\psi}\right) w(2) \\
&\quad +... + \left(\frac{\gamma^k}{1-\gamma\psi} - \frac{\gamma^k\psi}{1-\gamma\psi}\right) w(k) + ...
\end{aligned}
$$

The mean statistic of the multi-stage distribution $\mu(D)$ is derived as follows:

$$
\begin{aligned}
\mu(D) &= \Big(\frac{1}{1-\gamma\psi} + \frac{\gamma(1-\psi)}{1-\gamma\psi} + \frac{\gamma^2(1-\psi)}{1-\gamma\psi} + ... + \frac{\gamma^k(1-\psi)}{1-\gamma\psi} + ...\Big)\mu \\
&= \frac{1}{1-\gamma\psi}\Big(1 + \gamma(1-\psi) + \gamma^2(1-\psi) + ... + \gamma^k(1-\psi) + ...\Big)\mu \\
&= \frac{1}{1-\gamma\psi}\Big((1+\gamma+\gamma^2+...+\gamma^k+...) - \psi(\gamma^2+\gamma^3+...+\gamma^k+...)\Big)\mu \\
&= \frac{1}{1-\gamma\psi}\Big(\sum_{t=0}^{\infty}\gamma^t - \psi\sum_{t=1}^{\infty}\gamma^t\Big)\mu \\
&= \frac{1}{1-\gamma\psi}\Big(\frac{1}{1-\gamma} - \psi(\frac{\gamma}{1-\gamma})\Big)\mu \\
&= \frac{1}{1-\gamma\psi}\Big(\frac{1-\gamma\psi}{1-\gamma})\Big)\mu \\
&= \frac{1-\psi\gamma}{(1-\gamma)(1-\psi\gamma)}\mu \\
&= \frac{1}{(1-\gamma)}\mu
\end{aligned}
$$

The variance of the multi-stage distribution $Var(D)$ is given by the following summation:

$$
\begin{aligned}
Var(D) &= \left(\left(\frac{1}{1-\gamma\psi}\right)^2 + \left(\frac{\gamma(1-\psi)}{1-\gamma\psi}\right)^2 + \left(\frac{\gamma^2(1-\psi)}{1-\gamma\psi}\right)^2 + ... + \left(\frac{\gamma^k(1-\psi)}{1-\gamma\psi}\right)^2 + ...\right)\sigma^2 \\
&= \left(\frac{1+(1-\psi)^2(\gamma^2+\gamma^4+...+\gamma^{2k})+...}{(1-\gamma\psi)^2}\right)\sigma^2 \\
&= \left(\frac{1+(1-\psi)^2}{(1-\gamma\psi)^2\sum_{t=1}^{\infty}\gamma^{2t}.}\right)\sigma^2 \\
&= \left(\frac{1+(1-\psi)^2\big(\frac{\gamma^2}{1-\gamma^2}\big)}{(1-\gamma\psi)^2}\right)\sigma^2
\end{aligned}
$$

For $\psi = 0 \rightarrow Var(D) = \frac{1}{1-\gamma^2}\sigma^2$, which is same as the variant expression derived for problem statement 1. For $\psi = 1 \rightarrow Var(D) = \frac{1}{(1-\gamma)^2}\sigma^2$.

Therefore the multi-stage mean-$CVaR_\eta$ tradeoff becomes:

$$\begin{aligned}\lambda^*\mathbb{E}(D) + (1-\lambda^*)CVaR_\eta(D) &= \lambda^*\mathbb{E}(D) + (1-\lambda^*)(\mathbb{E}(D) - g(\eta)\frac{\sqrt{2Var(D)}}{2}) \\ &= \mathbb{E}(D) - (1-\lambda^*)g(\eta)\frac{\sqrt{2}}{2}\sqrt{Var(D)} \\ &= \frac{1}{(1-\gamma)}\mu - (1-\lambda^*)g(\eta)\frac{\sqrt{2}}{2}\sqrt{Var(D)}\end{aligned}$$

We next evaluate the sum of the proposed stage wise utility function over an infinite horizon:

$$\begin{aligned}\sum_{t=0}^{\infty}\gamma^t(\lambda_1\mathbb{E}(f_t) + (1-\lambda_1)CVaR_\eta(f_t)) &= \sum_{t=0}^{\infty}\gamma^t(\lambda_1\mathbb{E}(f(t)) + (1-\lambda_1)(\mathbb{E}(f(t)) - g(\eta)\sigma_I(f(t)) \\ &= \sum_{t=0}^{\infty}\gamma^t(\mathbb{E}(f(t)) - (1-\lambda_1)g(\eta)\sigma_I(f(t)) \\ &= \sum_{t=0}^{\infty}\gamma^t\mathbb{E}(f(t)) - (1-\lambda_1)g(\eta)\sum_{t=0}^{k}\gamma^t\sigma_I(f(t))\end{aligned}$$

Since $\sigma_I(f(t)) = \frac{\sqrt{2}}{2}Var(f(t))$, we evaluate the expressions for $Var(f(t))$ as follows

$$\begin{aligned}Var(f(0)) &= \sigma^2 \\ Var(f(1)) &= \sigma^2\left(\psi^2 + (1-\psi)^2\right) \\ Var(f(2)) &= \sigma^2\left(\psi^4 + (1-\psi)^2(1+\psi^2)\right) \\ &\cdot \quad \cdot \quad \cdot \\ VaR_(f(k)) &= \sigma^2\left(\psi^{2k} + (1-\psi)^2(1+\psi^2+\psi^4+...+\psi^{(2k-1)})\right) \\ &\cdot \quad \cdot \quad \cdot \\ &\cdot \quad \cdot \quad \cdot \\ &\cdot \quad \cdot \quad \cdot\end{aligned} \tag{88}$$

The evaluation $\sum_{t=0}^{\infty}\gamma^t\mathbb{E}(f(t))$ given the proposed objective coincides with $\mu(D) = \frac{1}{1-\gamma}\mu, \forall\gamma, \psi$

From Eq.(88), the evaluation of the $(1-\lambda_1)g(\eta)\sum_{t=0}^{\infty}\gamma^t\sigma_I(f(t))$ follows:

$$\begin{aligned}(1-\lambda_1)g(\eta)\sum_{t=0}^{\infty}\gamma^t\sigma_I(f(t)) &= (1-\lambda_1)g(\eta)\frac{\sqrt{2}}{2}\sigma\left(1+\gamma\left(\sqrt{\psi^2+(1-\psi)^2}\right)\right.\\ &\quad +\gamma^2\left(\sqrt{\psi^4+(1-\psi)^2(1+\psi^2)}\right)\\ &\quad +\gamma^3\left(\sqrt{\psi^6+(1-\psi)^2(1+\psi^2)+\psi^4}\right)+...+\\ &\quad \left.\gamma^k\left(\sqrt{\psi^{2k}+(1-\psi)^2(1+\psi^2+\psi^4+...+\psi^{2(k-1)})}\right)+...\right.\\ &= (1-\lambda_1)g(\eta)\sigma\frac{\sqrt{2}}{2}\Pi_1(\gamma,\psi)\end{aligned}$$

, where $\Pi_1(\gamma,\psi)=\sum_{l=0}^{\infty}\gamma^l\sqrt{\psi^{2l}+(1-\psi)^2\sum_{i=0}^{l-1}\psi^{2i}}$.

The mean statistic of the $D$ distribution coincides with the mean statistic of the proposed surrogate objective. In other words, these two objectives evaluate the same on the expected mean, but the main difference lies in the weight assigned to the standard deviation.

Similar to problem statement 2, we can derive an analytic expression for which $\lambda_1 = f(\lambda^*,\gamma,\psi)$. The expression follows:

$$\lambda_1 = 1-(1-\lambda^*)\frac{\sqrt{Var(D)}}{\Pi_1(\gamma,\psi)}$$

If we set $\lambda^*$=0 at the previous expression then:

$$\lambda_1 = 1-\frac{\sqrt{Var(D)}}{\Pi_1(\gamma,\psi)}$$

Trivially $\frac{\sqrt{Var(D)}}{\Pi_1(\gamma,\psi)}>0$, since $Var(D)$ and $\Pi_1(\gamma,\psi)>0$.

Similar to the last 2 cases, we need to show that $\frac{\sqrt{Var(D)}}{\Pi_1(\gamma,\psi)}<1$. This proof is in similar spirit with the last two problems. By assuming that $\frac{\sqrt{Var(D)}}{\Pi_1(\gamma,\psi)}<1$ is true, if we end up in a true statement then our initial assumption will be true:

$$\begin{aligned} \frac{\sqrt{Var(D)}}{\Pi_1(\gamma,\psi)} &< 1 \\ \sqrt{Var(D)} &< \Pi_1(\gamma,\psi) \\ Var(D) &< \Pi_1(\gamma,\psi)^2 \end{aligned} \tag{89}$$

Eq.(89) can be validated analytically for any $\gamma, \psi \in [0,1]$. Here, we display numerical results as shown in Fig.40 that correlate the parameters $\lambda^*$ with $\lambda_1$ for any $\gamma, \psi$. Given this parameter correlation, for this problem statement, the summation of the single period mean-CVaR results to an equivalent objective with the exact multistage mean - CVaR.

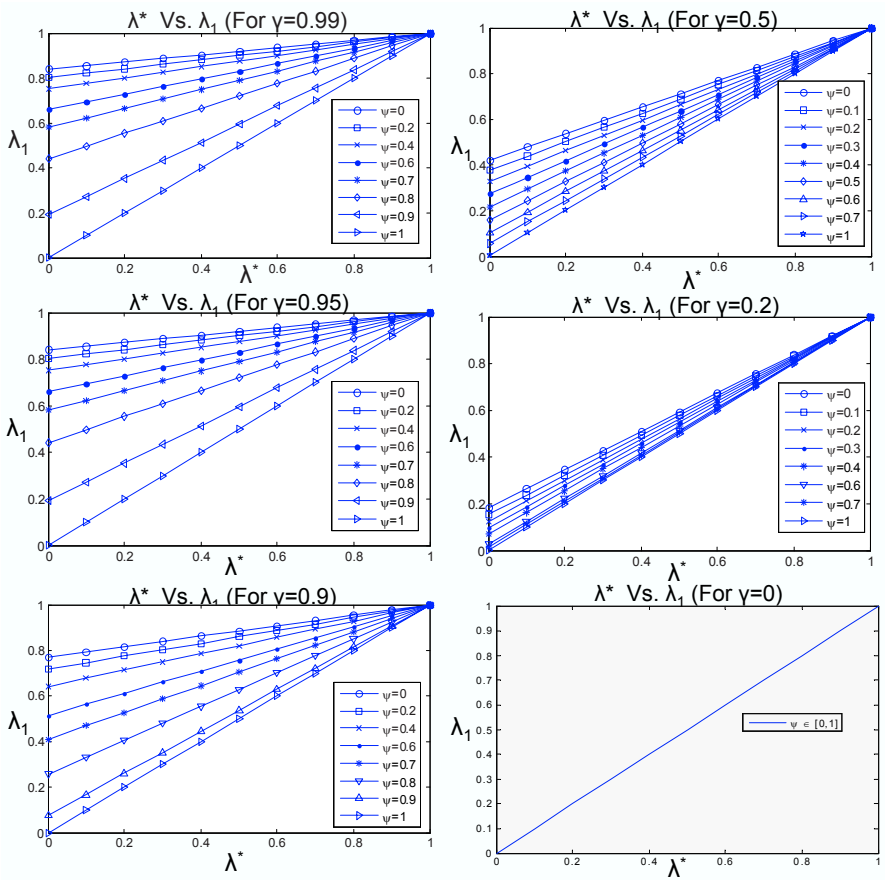



**Figure 40:** For problem statement 3, this figure demonstrates the correlation of the parameters $\lambda^*$ and $\lambda_1$ that makes the proposed objective and the exact multistage mean - CVaR equivalent for $\gamma$ and $\psi$ values.

## 6.5 A Proposed Algorithm That Approximates The Multi-stage Mean-CVaR Efficient Frontier For GDMDP's

The description of the proposed algorithm is tailored for GDMDP problems with a single goal state. Nonetheless, one can generalize this approach and apply it to problems where the cardinality of the set of goal states $\mathbb{S}_{\mathbb{G}}$ is more than one.

The necessary requirements for the solution of such problems are: **a)** the finiteness of the state space $\mathbb{S}$, **b)** a specified set of goal states $\mathbb{S}_{\mathbb{G}}$, **c)** a finite action space $\mathbb{A}$, **d)** a probability space $\Omega$ with finite support and **e)** the assumption that the state space is a network of states where each state can be reached from another state with positive probability.

**Step 1:** Pick a discount factor $\gamma < 1$.

**Step 2:** Solve via exact DP the problem using $\lambda_1 = 1$.

- Test the derived policy for $\lambda_1 = 1$, if it does not force the system to visit the goal in a finite number of steps, then go back to step 1 and increase the value of $\gamma$.
- If the derived policy does force the system to visit the goal state, go to step 3.

**Step 3:** Decrease the risk averse parameter $\lambda_1$ by 0.01 until $\lambda_1 = 0$ and solve the exact DP. Remember this parameter determines the weights between the single stage expected mean and $CVaR_\eta$ for each state action pair
$U_S(s_t, \alpha_t, \lambda_1) = \lambda_1 \mathbb{E}(f_t(s_t, \alpha_t)) + (1 - \lambda_1) CVaR_\eta(f_t(s_t, \alpha_t))$

- For each $\lambda_1$, test the derived policy via Monte Carlo simulations. If you find a $\lambda_1$ that does not force the system to visit the goal in a finite number of steps, then we have successfully determined the range of $\lambda_1$'s for which this choice of the discount factor allows you to get a valid policy that leads the system from the start to the goal state.
- If we are not satisfied with the range of $\lambda_1$'s that yield a valid policy, go to step 1 and increase the value of $\gamma$.

**Step 4:** Test all the policies derived for the choice of discount factor and the range of $\lambda_1$'s

via Monte Carlo simulations. Derive the corresponding efficient frontier. If satisfied exit algorithm.

- Up to this point, we should of have created an efficient frontier with at least 2 policies. One corresponding to $\lambda_1 = 1$, which is a risk neutral policy. The other policy may be the same for more than one values of $\lambda_1$. If the second policy does not belong to the efficient frontier, one should either go to step 1 and increase the discount factor $\gamma$ and simultaneously at step 3 decrease the increment 0.01 and repeat the entire procedure.

As seen in section 6.4, given our approach, if we decide to use a high value (e.g. 0.99) for the discount factor, the range of the risk averse parameter $\lambda_1$ that approximates the exact multi stage mean-CVaR trade-off will be close to 1.

Determining the exact range of the $\lambda_1$ parameter in a systematic manner is subject of further research. Note, that the nature of the algorithm does not change for large scale problems, one instead of using exact DP solutions will utilize ADP solutions.

### 6.5.1 Motivation Of Our Numerical Studies

The motivation of our numerical experiments is to study the mean-CVaR pareto efficiency in multi-stage problems by applying our proposed approach. Moreover, we will demonstrate the proposed approach within the ADP framework as presented at the previous chapter.

## *6.6 Deriving Efficient Frontier Solutions Using The Proposed Approach On Shortest Path Instances With Deterministic Transitions*

At this section we focus on deriving pareto optimal solutions on a 77 and 900 discrete state space shortest path examples with deterministic transitions.

In these numerical experiments, we tested two types of single period utilities which are: **a)** the discounted summation of single stage mean-variance Langragian utilities, **b)** the discounted summation of single stage mean-conditional value at risk Langragian utilities. We also tested a multi period utility chosen from the class of the exponential functions.

### 6.6.1 Shortest Path With 77 Discrete States: Problem Description

We will be addressing shortest path problems a common objective is to minimize the expected cost along with the expected cost associated with the $\eta \times 100\%$ worst cases. (Usually, for cost distributions $\eta$ is set to 0.95 and not to 0.05 as assigned for profit distributions). By assuming that each reward is a negative quantity except from the ones generated when the system is at goal state which is zero, our formulation is equivalent to minimizing the expected cost with positive costs and the $CVaR_\eta$ with respect to a cost distribution. For the next sections, we adopt the definition that $CVaR_\eta$ represents the expected cost associated with the 5% of the worst cases. In *Rockafellar and Uryasef* [50], the authors formally define $CVaR_\eta$ over a cost distribution.

Therefore for the rest of the chapter, we apply $CVaR_\eta$ on a myopic conditional cost distribution $f(s_{t+1}|s_t, \alpha_t)$ as a mean to generate policies and on the summation of a stream of rewards as a mean to evaluate this multi stage risk measure. Assuming that $f(s_{t+1}|s_t, \alpha_t) \sim N(\mu, \sigma)$ the $CVaR_\eta f(s_{t+1}|s_t, \alpha_t)$ is quantified by

$$CVaR_\eta(f(s_{t+1}|s_t, \alpha_t)) = \mu + g(\eta)\sigma$$

, where $g(\eta) = \sqrt{2}\exp(erf(2\eta - 1)^2)^{-1}(1 - \eta)^{-1}$ and $\mu, \sigma \geq 0$.

The cost structure of the problem of interest appears in Fig.41. The state space of this problem can be effectively described by a one dimensional vector, since we can enumerate all the positions at the x-y plane. For instance the position on the graph $(0, 0)$ is numbered as state 1, the position $(0, 1)$ is numbered as state 2 ,..., the position $(0, 6)$ is numbered as state 7, the position $(1, 0)$ is numbered as state 8,...,the position $(10, 6)$ is numbered as state 77. The action space includes moves only to neighboring positions (excluding those reachable via diagonal moves). The system moves with probability **1** in the corresponding direction. Specifically the starting state is the position $(0, 0)$, while the end state is the position $(10, 6)$. One incurs a normally distributed cost $g(s_t) \sim N(\mu_{s_t}, \sigma_{s_t})$ based on the state visited. Our goal is to find the path from (0,0) to (10,6), that not only minimizes the expected cost but also minimizes the expected cost associated with the $5 \times 100\%$ worst cases.

### 6.6.2 Applying The Approach Using As Single period Utilities: A) The Mean-CVaR Function B) The Mean-Variance Function

Here, we denote the risk averse parameter with respect to the single period mean-CVaR (mean-VaR) as $\lambda_1(\lambda_2)$. What follows is the application of the proposed approach to this instance, when we adopt the single period mean-CVaR trade-off:

**Step 1:** Pick $\gamma = 0.99$.

**Step 2:** We use $\lambda_1 = 1$ and solve the exact DP .

- It lead the system to the goal in a finite number of steps.

**Step 3:** Decrease the risk averse parameter $\lambda_1$ by 0.01 until $\lambda_1 = 0$ and solve the exact DP. Remember this parameter determines the weights between the single stage expected cost and the risk measure $CVaR_\eta$ for each state action pair $U_S(s_t, \alpha_t, \lambda_1) = \lambda_1 \mathbb{E}(f_t(s_{t+1}|s_t, \alpha_t)) + (1 - \lambda_1) CVaR_\eta(f_t(s_{t+1}|s_t, \alpha_t))$ . The $\eta$ used is 0.95.

- The range of $\lambda_1$ for which we can recover valid policies is $\lambda_1 = [0, 1]$.

**Step 4:** We test the derived policies with Monte Carlo simulations. The cumulative results-efficient frontier solutions for this single stage problem appear at Table 18. We are satisfied with the resulting efficient frontier and we exit the algorithm.

From Table 18, we observe that the summation of the mean-CVaR or the mean-VaR Langragian single period objectives yield the same spectrum of solutions.

*6.6.2.1 Evaluating The Objectives:A)* $\mathbb{E}\big[\sum_t CVaR_{0.95}(U_S(s_t, \alpha_t, \lambda_1)\big]$ *B)* $\mathbb{E}\big[CVaR_{0.95}(\sum_t (U_S(s_t, \alpha_t, \lambda_1))\big]$

In Fig.42, we display the optimal trajectories for $\lambda_1 = 1$,$\lambda_1 = 0.95$,$\lambda_1 = 0$.

For $\lambda_1 = 1$, the solution corresponds to the minimum expected cost ( $\mathbb{E}\big[\sum_t (U_S(s_t, \alpha_t, \lambda_1 = 1))\big]$=269 ) and to a corresponding multistage $CVaR_{0.95}$ measure, which is $\mathbb{E}\big[CVaR_{0.95}(\sum_t U_S(s_t, \alpha_t, \lambda_1 = 1))\big]$=352.

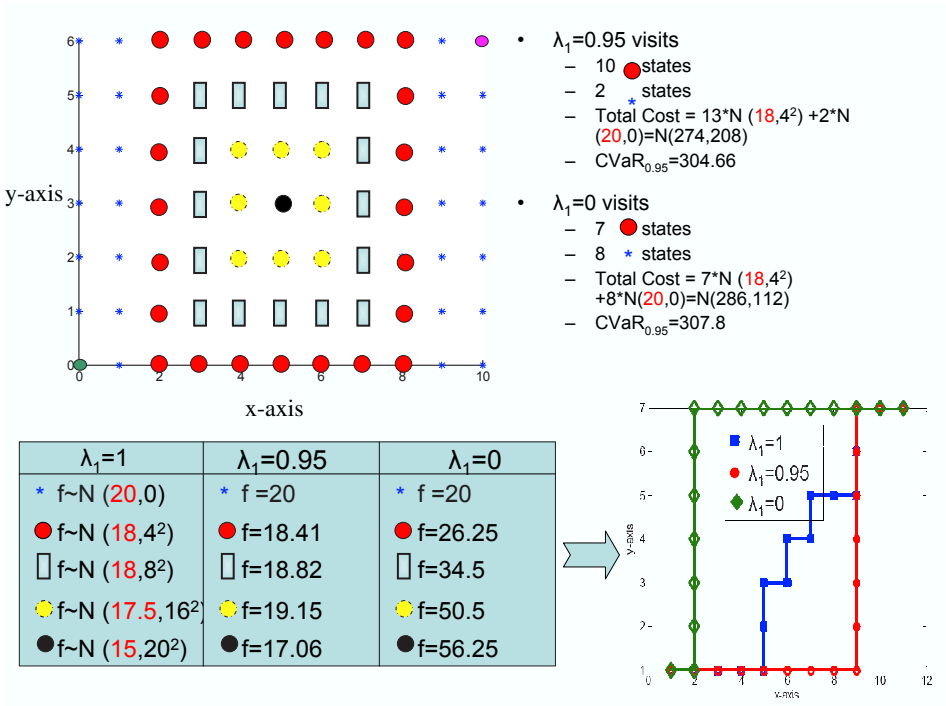



**Figure 41:** Schematic illustration of the cost structure of the one dimensional shortest path problem with the 77 discrete states. We display the optimal routes for $\lambda_1 = 0, \lambda_1 = 0.95, \lambda_1 = 1$ and show explicitly why minimizing the summation of individual $CVaR$ will not necessarily minimize the multistage $CVaR$.

In Table 18, we observe that the policy which yields the minimum multistage $CVaR_{0.95}$ measure corresponds to a range of $\lambda_1$ values which is $\lambda_1 = [0.75, 0.95]$ and not to $\lambda_1 = 0$. This comes in agreement with the intuition provided by our analytical results as shown in section 6.4.

To validate the correctness of this result, we form Tables 19,20, where we evaluate that the corresponding solutions for $\lambda_1 = 0.95$ and $\lambda_1 = 0$ are optimal with respect to the transformed, for $\lambda_1 = 0.95$ and $\lambda_1 = 0$, cost structure.

From Table 18 and Table 20, we can quantify that the policy produced when minimizing $\mathbb{E}\big[\sum_t (U_S(s_t, \alpha_t, \lambda_1 = 0))\big]$ will not necessary minimize $\mathbb{E}\big[CVaR_{0.95} \sum_t (U_S(s_t, \alpha_t, \lambda_1 = 0))\big]$.

From these results, we can conclude that minimizing the summation of single stage conditional value at risk costs is not equivalent with minimizing the conditional value at risk of the summation of single stage costs. This agrees with our analytical results, since the generated solutions for $\lambda_1 = 0$ apparently do not belong to the multistage efficient frontier.

**Table 18:** For this shortest path problem with the deterministic transitions ($\mathbf{p} = 1$) this table demonstrates: the statistics that correspond to the policies derived from the parametric summation of the single stage mean-CVaR and single stage mean-Variance tradeoff. For each policy we demonstrate its mean $\mu$ and the evaluation of the multi stage risk measures ($\sigma$,$CVaR_{0.95}$,$VaR_{0.95}$ ) associated with it.

| Risk Averse Parameter | $\mu$ | $\sigma$ | $VaR_{0.95}$ | $CVaR_{0.95}$ |
|---|---|---|---|---|
| $\lambda_1 \in [0.98 - 1]$ - $\lambda_2 = [0.97 - 1]$ | 269 ± 1.2 | 40.5 ± 0.8 | 335.3 ± 4.5 | 352 ± 5.6 |
| $\lambda_1 = [0.96 - 0.97]$ - $\lambda_2 = [0.93 - 0.96]$ | 270 ± 1 | 34.1 ± 0.8 | 326.1 ± 2.1 | 340.4 ± 2.5 |
| $\lambda_1 = [0.76 - 0.95]$ - $\lambda_2 = [0.67 - 0.92]$ | 274 ± 0.5 | 14.4 ± 0.4 | 297.6 ± 1 | 303.7 ± 1.1 |
| $\lambda_1 = [0 - 0.75]$ - $\lambda_2 = [0.1 - 0.66]$ | 286 ± 0.3 | 10.6 ± 0.2 | 303.3 ± 0.7 | 307.7 ± 0.9 |

**Table 19:** Given the transformed shortest path problem with the deterministic transitions ($\mathbf{p} = 1$), where the myopic cost is described by $(U_S(s_t, \alpha_t, \lambda_1 = 0.95)) = 0.95\mu_{s_t} + 0.05CVaR_{0.95}f(s_t, \alpha_t)$. This table demonstrates the expected performance of each produced policy for $\lambda_1 = 1$,$\lambda_1 = 0.95$,$\lambda_1 = 0$ evaluated at the corresponding objective: $\mathbb{E}\big[\sum_t (U_S(s_t, \alpha_t, \lambda_1 = 0.95))\big]$.

| Policies with respect to the original cost structure corresponding to | $\mathbb{E}\big[\sum_t (U_S(s_t, \alpha_t, \lambda_1 = 0.95))\big]$ |
|---|---|
| $\lambda_1 = 1$ | 281.79 |
| $\lambda_1 = 0.95$ | 279.36 |
| $\lambda_1 = 0$ | 343.76 |

**Table 20:** Given the transformed shortest path problem with the deterministic transitions ($\mathbf{p} = 1$), where the myopic cost is described by $(U_S(s_t, \alpha_t, \lambda_1 = 0)) = CVaR_{0.95}f(s_t, \alpha_t)$. This table demonstrates the expected performance of each produced policy for $\lambda_1 = 1$,$\lambda_1 = 0.95$,$\lambda_1 = 0$ evaluated at the corresponding objective: $\mathbb{E}\big[\sum_t (U_S(s_t, \alpha_t, \lambda_1 = 0))\big]$.

| Solutions from the Original Cost Matrix corresponding to | $\mathbb{E}\big[\sum_t (U_S(s_t, \alpha_t, \lambda_1 = 0))\big]$ |
|---|---|
| $\lambda_1 = 1$ | 524.77 |
| $\lambda_1 = 0.95$ | 381.26 |
| $\lambda_1 = 0$ | 343.76 |

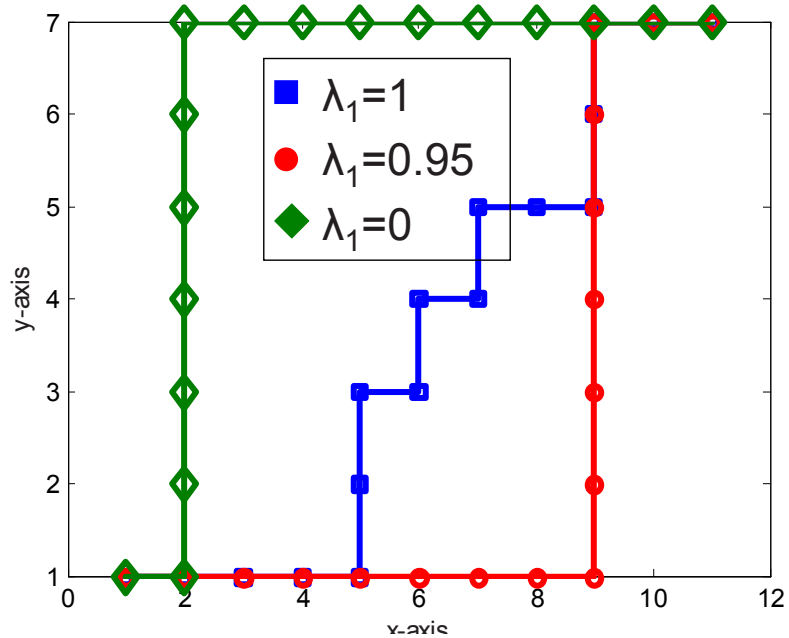



**Figure 42:** Schematic illustration of the optimal solutions for $\lambda_1 = 1$,$\lambda_1 = 0.95$,$\lambda_1 = 0$.

### 6.6.3 Applying The Multi stage Exponential Utility On The Shortest Path Problem

In order to use the the exponential utility to derive risk averse solution, we need to transform the non-positive rewards to the corresponding utilities and then pick a risk averse parameter $0 < \xi < 1$.

Theorem 6.2.2.a of *Putterman* [11] provides the basis for a primal linear programming formulation. The LP formulation for the exponential inter-period utility is the following:

$$\begin{aligned} &Minimize && \textstyle\sum_{s_t \in \mathbb{S}} b_(s_t) J_M(s_t) \\ &s.t. && J_M(s_t) - \textstyle\sum_{s_{t+1} \in \mathbb{S}} P(s_{s+1}|s_t, \alpha_t) \xi^{f(s_t+1|s_t,\alpha_t)} J_M(s_{t+1}) \geq 0, \forall s_t \in \mathbb{S}, \forall \alpha_t \in \mathbb{A} \end{aligned} \tag{90}$$

where the $b(s_t)$ arbitrary positive coefficients that must sum to 1: $\sum_{s_t \in \mathbb{S}} b_(s_t) = 1$.

For the above problem, the constraints form a coefficient matrix A. The condition number of the tall matrix A (308 rows and 77 columns) as a function of the risk parameter $\xi$ is displayed at Fig.43. Briefly, the condition number is a measure of stability or sensitivity of a matrix (or the linear system it represents) to numerical operations. In other words, we may not be able to trust the results(inverse) of an ill-conditioned matrix. In this case as $\xi \to 0$

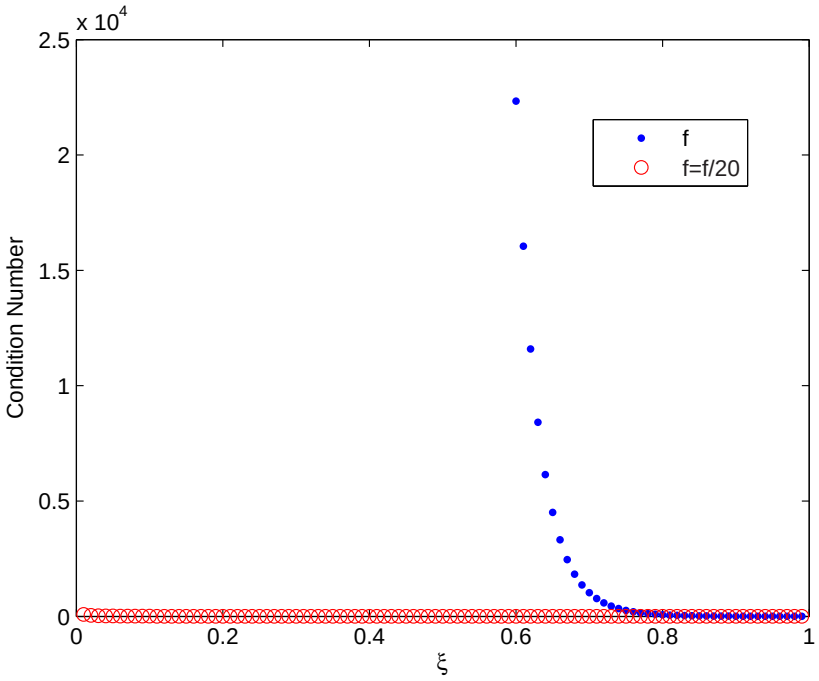



**Figure 43:** Solving the optimality equations for the 77 discrete state shortest path problem when using the multi period utility. This plot shows the condition number of the corresponding linear program as a function of the parameter $\xi$.

the condition number increases exponentially. Only matrices with condition numbers near 1 are said to be well-conditioned. A simple way to fix the condition number is to normalize the A matrix by multiplying all the rewards by 1/20 (Fig.43), but then one changes the nature of the problem.

Its obvious that because of the condition number this multi period utility can yield meaningful solutions only for certain values of $\xi$'s. This inter-period utility can reveal only two meaningful policies for this problem for $\xi$'s ranging from 0.75 to 1. The retrieved policies match the first two policies as shown in Table 18. This approach is unable to construct the other two policies.

In summary, this result for this seemingly easy problem indicates that the usage of multi period exponential utilities is numerically unstable and cannot yield the full available spectrum of risk averse policies.

### 6.6.4 Deriving Efficient Frontier Solutions On 900 Discrete State Shortest Path With Deterministic Transitions

The cost data for this instance are available for download from the following website http://www.chbe.gatech.edu/lee/members /npratikakis.shtml. These data are in the form of matlab files. We also illustrate the data in Fig.44.

In order to capture correctly the time-risk tradeoff in a given GDMDP we need to set wisely the discount factor. For this problem we set $\gamma$ to be 0.995. This determines that the optimal value function will account for 200 future moves.

What follows is the application of our approach to this problem.

**Step 1:** Pick $\gamma = 0.995$.

**Step 2:** We solve the exact DP using $\lambda_1 = 1$.

- The produced policy leads the system to the goal in a finite number of steps.

**Step 3:** Decrease the risk averse parameter $\lambda_1$ by 0.01 until $\lambda_1 = 0$ and solve the exact DP.

- The range of $\lambda_1$ for which we can recover valid policies is $\lambda_1 = [0.93, 1]$.

**Step 4:** Using Monte carlo simulation, we verify that this range of $\lambda_1$ creates 3 policies that belong to the multistage mean-CVaR efficient frontier. The most risk averse policy is the one, which corresponds to the least multistage $CVaR_\eta$. This policy is produced when $\lambda_1 = 0.98$.

In order to create additional policies we would need to go go to step 1 and increase the discount factor $\gamma$ and simultaneously at step 3 and manipulate the increment 0.01. Here, we let $\gamma$ unchanged but we do manipulate the increment of change for the $\lambda_1$ parameter from 0.01 to 0.001. For $\lambda_1 = 0.998$, we retrieve the policy with the characteristics as shown in Table 21.

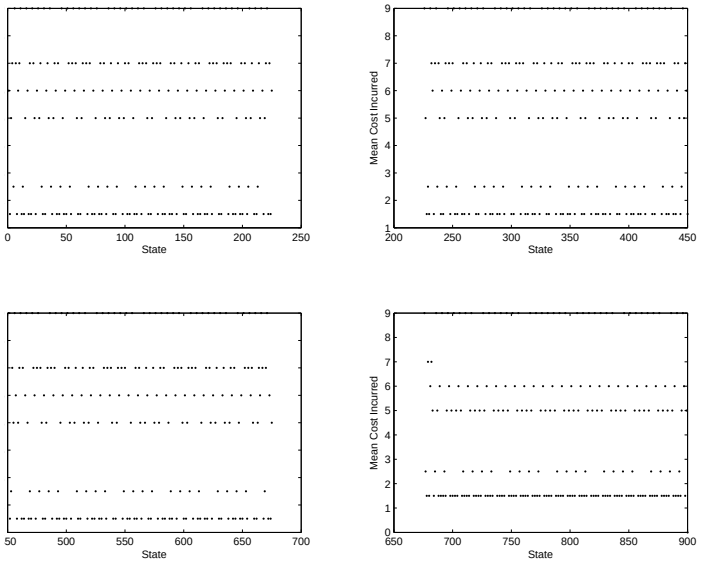
**Step 5:** Hence, we have derived 4 policies that belong to the multistage mean-CVaR efficient frontier and we exit the algorithm.

**Table 21:** For this shortest path problem with the deterministic transitions ($\mathbf{p} = 1$) this table demonstrates: the statistics that correspond to the policies derived from the parametric summation of the single stage mean-CVaR tradeoff. For each policy we demonstrate its mean $\mu$ and the evaluation of the multi stage risk measures ($\sigma$,$CVaR_{0.95}$,$VaR_{0.95}$ ) associated with it.

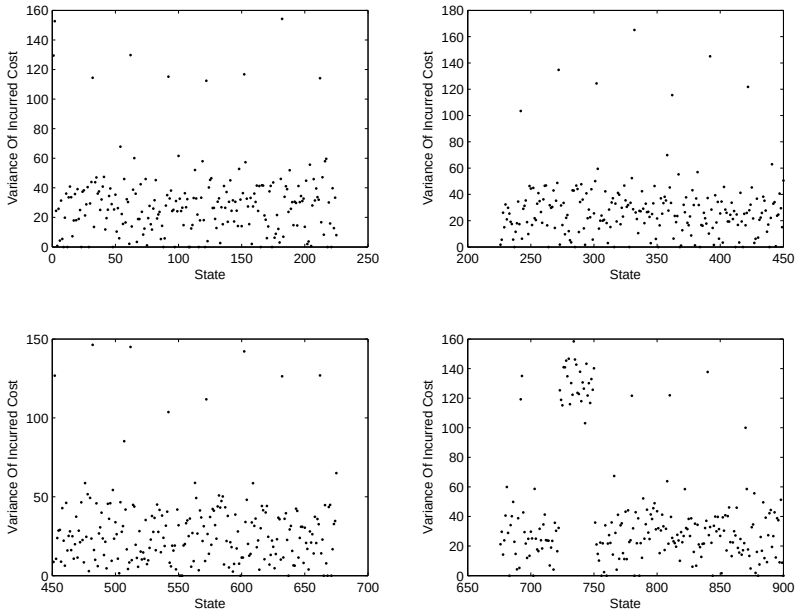
| Risk Averse Parameter | $\mu$ | $\sigma$ | $VaR_{0.95}$ | $CVaR_{0.95}$ | Steps To Reach Goal |
|---|---|---|---|---|---|
| $\lambda_1 = 1$ | 144 ± 22 | 724 ± 16 | 1,334 ± 42 | 1,635 ± 55 | 62 |
| $\lambda_1 = 0.998$ | 159 ± 11 | 307 ± 6 | 668 ± 17 | 796 ± 19 | 58 |
| $\lambda_1 = 0.99$ | 160 ± 9 | 280 ± 6 | 613 ± 16 | 734 ± 21 | 58 |
| $\lambda_1 = [0.93, 0.98]$ | 161 ± 8 | 258 ± 5 | 583 ± 16 | 692 ± 17 | 58 |

Note, that the mean values for the policies derived when: a) $\lambda_1 = 0.99$, b) $\lambda_1 = 0.998$, c) $\lambda_1 = 0.99$, and d) $\lambda_1 = 0.98$ are very close. In this application, we optimize mostly with respect to the variance.

If the multi period exponential utility is applied, we generate only a single policy which does not belong to the efficient frontier. Briefly, the characteristics of that policy are: a ) Its mean is $237 \pm 14.35$, b) its multistage variance is $434 \pm 10.13$, c) its multistage conditional value at risk $1,182 \pm 50$.




(a) We display the mean of each normal distribution that corresponds to the expected single stage cost incurred when visiting a particular state at a 30x30 Grid (900 discrete state example).




(b) We display the variance of each normal distribution that corresponds to the expected single stage cost incurred when visiting a particular state at a 30x30 Grid (900 discrete state example).

**Figure 44:** Illustration of cost data for the 900 discrete state stochastic shortest path problem.

In summary, to optimize the time-risk tradeoff in GDMDP's, the horizon effect can be captured by the proper choice of the discount factor and the risk effect by the proper choice of the risk averse parameter $\lambda_1$.

## *6.7 Approximating The Multi-stage Mean-CVaR Efficient Frontier Using The Proposed Approach*

In this section, we will address the same problems as the ones shown in the previous section with the additional complexity of the probabilistic transition rules. In the following problems, the system transitions with probability **p** in the corresponding action direction and with probability (1-**p**)/3 in a different direction.

We will derive results for the 77 discrete state space problem using both single period and multi period exponential utilities, when **p**={0.9,0.8}. We will also apply our approach on the 900 discrete state space problem for **p**={0.8}.

### 6.7.1 Applying The Approach On A 77 Discrete State Shortest Path With Probabilistic Transitions

Here, we delineate the proposed approach for $\mathbf{p} = 0.9$ and $\mathbf{p} = 0.8$ .

**Step 1:** Pick $\gamma = 0.99$.

**Step 2:** We solve the exact DP using $\lambda = 1$.

- The produced policy does lead the system to the goal state in a finite number of steps.

**Step 3:** Decrease the risk averse parameter $\lambda_1$ by 0.01 until $\lambda_1 = 0$ and solve for each $\lambda_1$ the exact DP.

- The range of $\lambda_1$ for which we can recover valid policies is $\lambda_1 = [0, 1]$

**Step 4:** Each produced policy is tested via Monte Carlo simulations. The cumulative results-solutions for these multi stage problem appear in Figure 45 and 46.

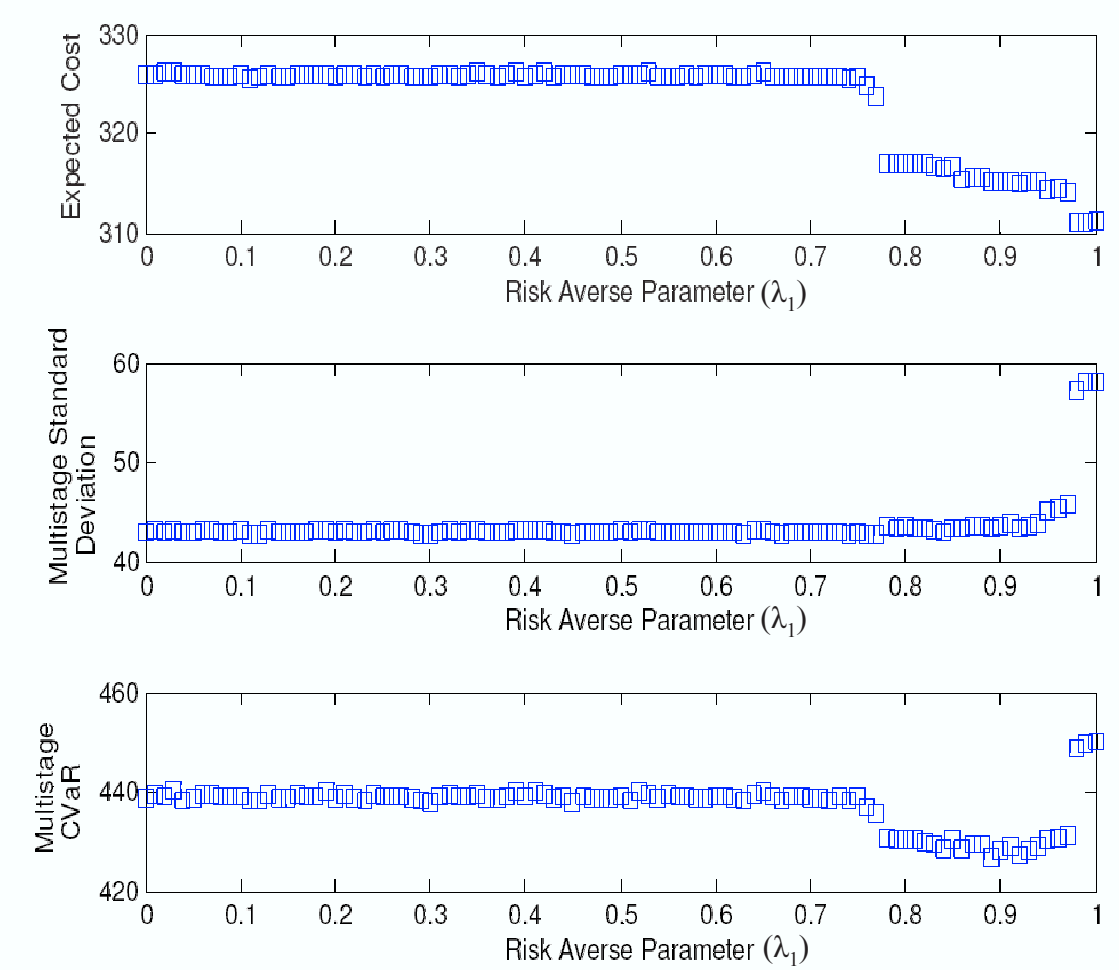



**Figure 45:** For this multistage shortest path problem with the probabilistic transitions($\mathbf{p} = 0.9$) this figure demonstrates: the performance and the corresponding multistage risk measures given the policies derived from DP, if we set as objective the parametric summation of the single stage mean-CVaR tradeoff (the risk averse parameter ranges from 0 to 1).

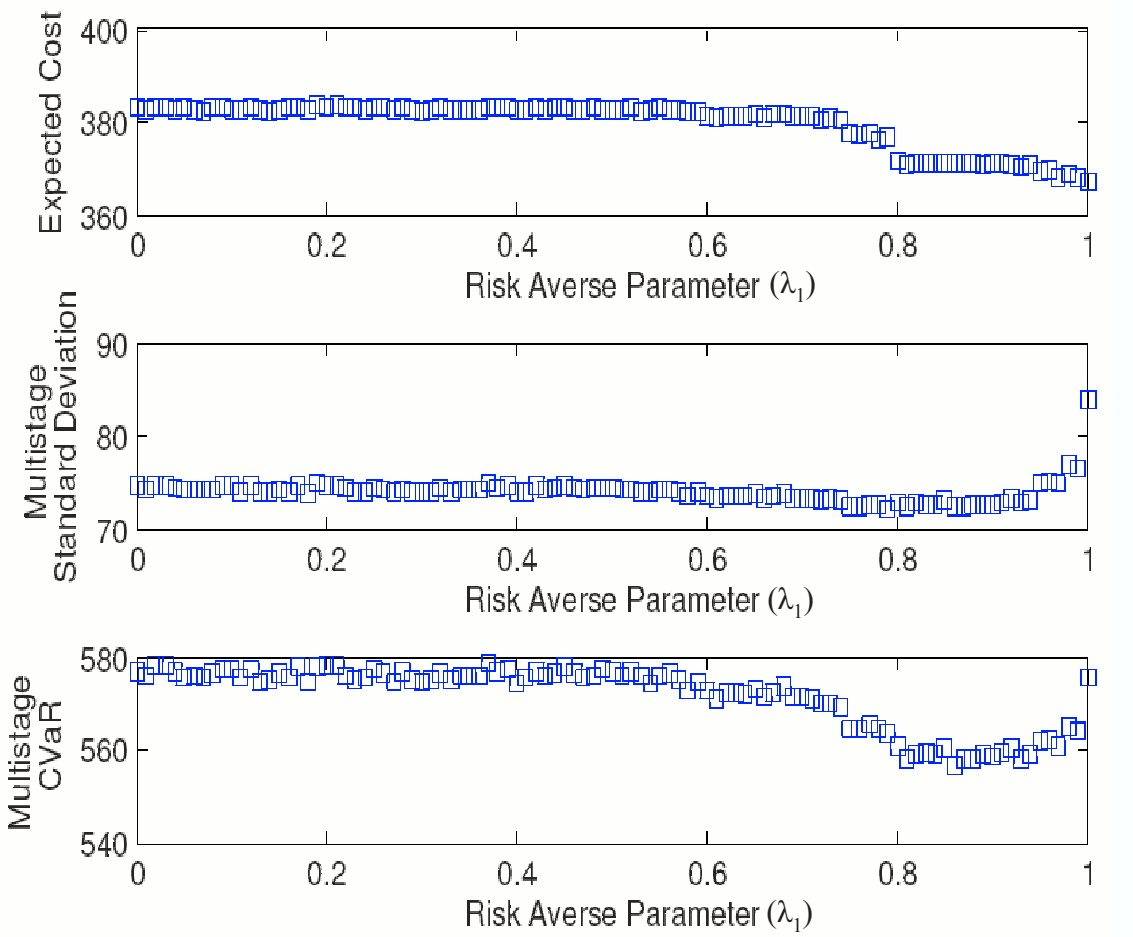



**Figure 46:** For this multistage shortest path problem with the probabilistic transitions($\mathbf{p} = 0.8$) this figure demonstrates: the performance and the corresponding multistage risk measures given the policies derived from DP, if we set as objective the parametric summation of the single stage mean-CVaR tradeoff (the risk averse parameter ranges from 0 to 1).

**Table 22:** Given this multi stage problem with the probabilistic transitions ($\mathbf{p} = 0.9$), this table demonstrates the expected performance of each DP solution, when for each state the myopic cost is transformed by $U_S(s_t, \alpha_t, \lambda_1) = \lambda_1 \mu(s_t) + (1 - \lambda_1) CVaR_{0.95}(f(s_t))$. We evaluated the DP policies for $\lambda_1 = 1_1$,$\lambda_1 = 0.89$,$\lambda_1 = 0$, on the following objective $\mathbb{E}\big[\sum_t (U_S(s_t, \alpha_t \lambda_1 = 0.89))\big]$.

| Policies with respect to the original cost structure corresponding to | $\mathbb{E}\big[\sum_t (U_S(s_t, \alpha_t, \lambda_1 = 0.89))\big]$ |
|---|---|
| $\lambda_1 = 1$ | 344.15 |
| $\lambda_1 = 0.89$ | 329.06 |
| $\lambda_1 = 0$ | 399.77 |

- The following results correspond to the 77 discrete state space problem with probabilistic transitions $\mathbf{p} = 0.9$.

  Each of the points on Figures 45,46 represent the statistics of a derived policy, which is evaluated 100 times via 10,000 Monte Carlo simulations, in order to derive numerical error bounds. Each of these simulation starts from the starting state (0,0) and ends at the goal state (10,6).

  For $\lambda_1 = 1$, the solution corresponds to the minimum expected cost ( $\mathbb{E}\big[\sum_t U_S(s_t, \alpha_t, \lambda_1 = 1)\big]$=311.3 ) and to a corresponding multistage $CVaR_{0.95}$ measure, which is $\mathbb{E}\big[CVaR_{0.95}(\sum_t U_S(s_t, \alpha_t, \lambda_1 = 1)\big]$=450.19.

  In Fig. 45, we observe that the policy which yields the minimum multistage $CVaR_{0.95}$ measure corresponds to $\lambda_1 = 0.89$ and not to $\lambda_1 = 0$. This is in correspondence with our theoretical analysis.

  To validate the correctness of this result for $\mathbf{p} = 0.9$, we form Table 22 and Table 23, where we evaluate that the corresponding solutions for $\lambda_1 = 0.89$ and $\lambda_1 = 0$ are optimal with respect to the transformed, for $\lambda_1 = 0.89$ and $\lambda_1 = 0$, cost structure.

  From Fig.45 and Table 22, we can quantify that the policy instructed by $\lambda_1 = 0$ that minimizes $\mathbb{E}\big[\sum_t U_S(s_t, \alpha_t, \lambda_1 = 0)\big]$ will not necessarily be the same policy that minimizes $\mathbb{E}\big[CVaR_{0.95} \sum_t U_S(s_t, \alpha_t, \lambda_1)\big]$.

**Table 23:** Given this multi stage problem ($\mathbf{p} = 0.9$), this table demonstrates the expected performance of each DP solution, when for each state the myopic cost is transformed by $f_t(s_t, \lambda_1) = \lambda_1 \mu(s_t) + (1 - \lambda_1) CVaR_{0.95}(f(s_t))$. We evaluated the DP policies $\lambda = 1_1$,$\lambda_1 = 0.89$,$\lambda_1 = 0$, on the following objective $\mathbb{E}\big[\sum_t U_S(s_t, \alpha_t, \lambda_1 = 0)\big]$.

| Solutions from the Original Cost Matrix corresponding to | $\mathbb{E}\big[\sum_t U_S(s_t, \alpha_t, \lambda_1 = 0)\big]$ |
|---|---|
| $\lambda_1 = 1$ | 612.05 |
| $\lambda_1 = 0.89$ | 441.24 |
| $\lambda_1 = 0$ | 399.76 |

- The following discussion correspond to the 77 discrete state space problem when $\mathbf{p} = 0.8$.

  For $\lambda_1 = 1$, the solution corresponds to the minimum expected cost ( $\mathbb{E}\big[\sum_t (f_t(s_t, \lambda_1 = 1))\big]$=366.97 ) and to a corresponding multistage $CVaR_{0.95}$ measure, which is $\mathbb{E}\big[CVaR_{0.95}(f_t(s_t, \lambda_1 = 1))\big]$=575.78.

  In Fig. 46, we observe that the policy which yields the minimum multistage $CVaR_{0.95}$ measure corresponds to $\lambda_1 = 0.86$ and not to $\lambda_1 = 0$. This is in correspondence with our theoretical analysis.

  To validate the correctness of this result for $\mathbf{p} = 0.8$, we form Table 25 and Table 26, where we evaluate that the corresponding solutions for $\lambda = 0.86$ and $\lambda = 0$ are optimal with respect to the transformed, for $\lambda = 0.86$ and $\lambda = 0$, cost structure.

  From Fig. 46 and Table 25, we can quantify that the policy instructed by $\lambda_1 = 0$ that minimizes $\mathbb{E}\big[\sum_t U_S(s_t, \alpha_t, \lambda_1 = 0)\big]$ will not necessary be the same policy that minimizes $\mathbb{E}\big[CVaR_{0.95} \sum_t (f_t(s_t, \lambda_1))\big]$.

**Table 24:** Given this multi stage problem ($\mathbf{p} = 0.9$), this table demonstrates the expected performance of each DP solution, when for each state the myopic cost is transformed by $f_t(s_t, \lambda_1) = \lambda_1 \mu(s_t) + (1 - \lambda_1) CVaR_{0.95}(f(s_t))$. We evaluated the DP policies $\lambda = 1_1$,$\lambda_1 = 0.86$,$\lambda_1 = 0$, on the following objective $\mathbb{E}\big[\sum_t U_S(s_t, \alpha_t, \lambda_1 = 0.86)\big]$ .

| Policies with respect to the original cost structure corresponding to | $\mathbb{E}\big[\sum_t U_S(s_t, \alpha_t, \lambda_1 = 0.86)\big]$ |
|---|---|
| $\lambda_1 = 1$ | 417.5 |
| $\lambda_1 = 0.86$ | 392.8 |
| $\lambda_1 = 0$ | 396 |

**Table 25:** Given this multi stage problem ($\mathbf{p} = 0.9$), this table demonstrates the expected performance of each DP solution, when for each state the myopic cost is transformed by $f_t(s_t, \lambda_1) = \lambda_1 \mu(s_t) + (1 - \lambda_1) CVaR_{0.95}(f(s_t))$. We evaluated the DP policies $\lambda = 1_1$,$\lambda_1 = 0.86$,$\lambda_1 = 0$, on the following objective $\mathbb{E}\big[\sum_t U_S(s_t, \alpha_t, \lambda_1 = 0)\big]$ .

| Solutions from the Original Cost Matrix corresponding to | $\mathbb{E}\big[\sum_t U_S(s_t, \alpha_t, \lambda_1 = 0)\big]$ |
|---|---|
| $\lambda_1 = 1$ | 689 |
| $\lambda_1 = 0.86$ | 524 |
| $\lambda_1 = 0$ | 483 |

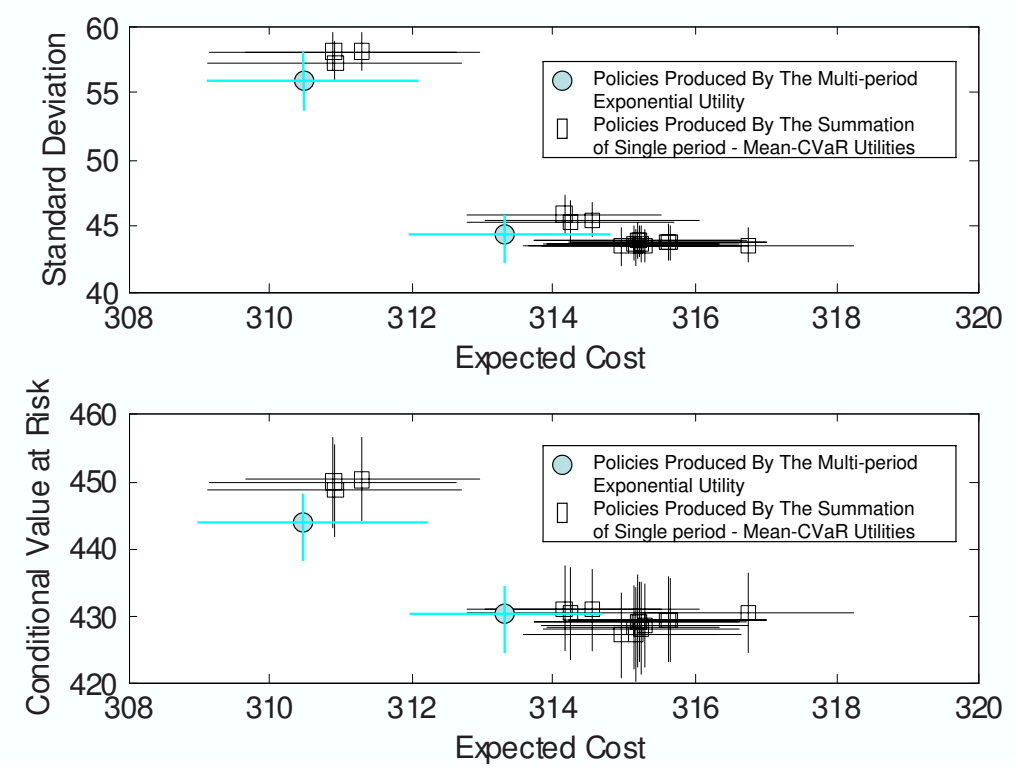



**Figure 47:** Resulting efficient frontiers, when using multi period and intra-period utilities for $\mathbf{p} = 0.9$.

- The discount factor was set to 0.99. Therefore, the horizon effect fades after 100 steps. On average starting from state 1, for $\mathbf{p}$=0.9(0.8) and $\lambda_1 = 1$, it takes 18.3(21.5) steps to reach the goal. This shows that for multi-stage systems with a medium horizon the originally proposed approach yields a good approximation of the exact multistage mean-CVaR efficient frontier

- The cumulative results when using the exponential interperiod utilities appear at Fig.47,48. In both cases ($\mathbf{p} = 0.9$,$\mathbf{p} = 0.8$ ), if $\xi$ is set to less than 0.7 the VI does not converge to a proper solution, even if the initial value obey the following necessary condition that ensures convergence $signln\gamma \geq J_M^0 \geq J_M^*$ ( [47]). The reason that VI does not converge is mainly related with the propagation of numerical errors due to the resulting condition number . In this case, the only way to perform VI is to uniformly scale in a heuristic fashion the single stage cost.

- The findings support the effectiveness of the proposed single period utility, since it

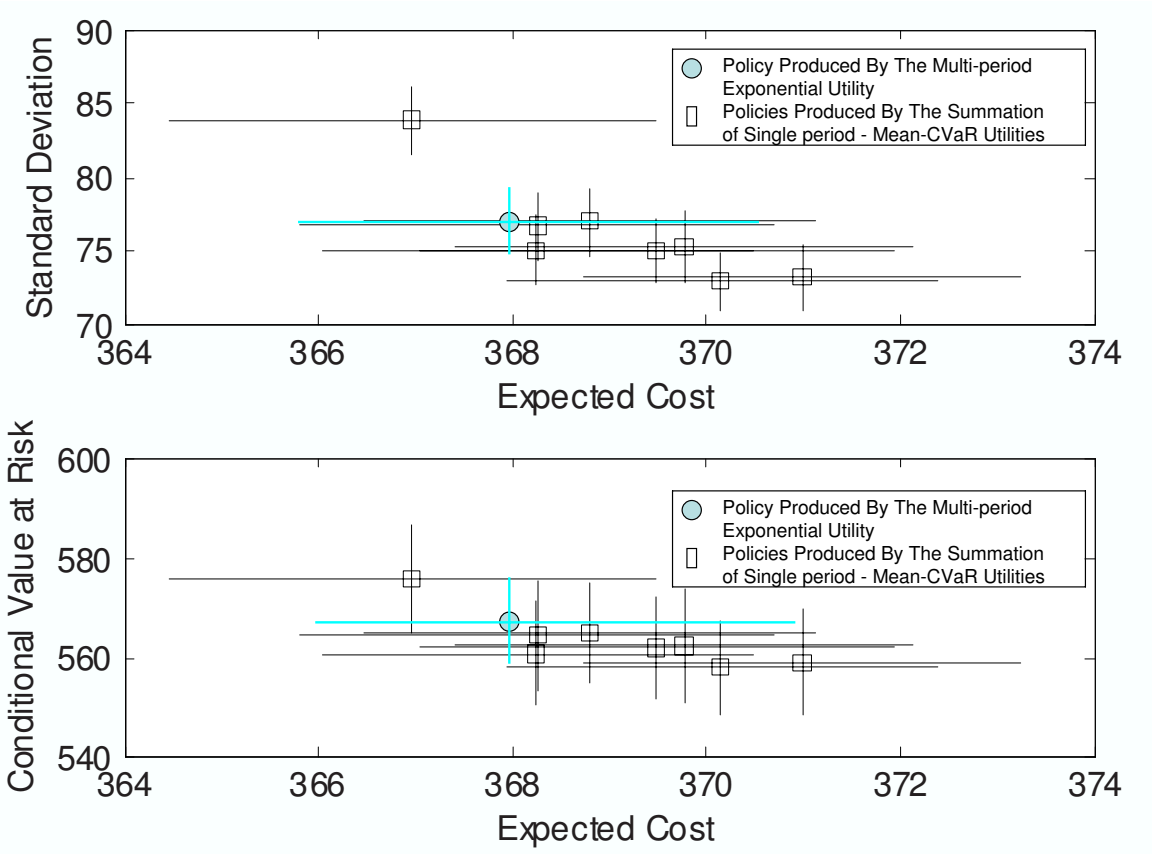



**Figure 48:** Resulting efficient frontiers, when using inter-period and intra-period utilities for $\mathbf{p} = 0.8$.

does yield multistage policies with respect to the risk measure $CVaR$. With the proposed approach we trace the effective range of $\lambda_1$ that yield valid policies, since when $\lambda_1$ goes to 0 the derived DP policies do not belong to an efficient frontier.

### 6.7.2 Results On 900 Discrete State Shortest Path Using The Proposed Approach.

In this section, we will apply the proposed approach to the 900 discrete state shortest path example with probabilistic transitions described by $\mathbf{p} = 0.8$.

**Step 1:** Pick $\gamma = 0.9$.

**Step 2:** We solve the exact DP using $\lambda_1 = 1$..

As discussed before the discount factor determines the horizon effect and therefore the whole nature of the multistage problem. For GDMDP's applications, when using

a discount factor we are in danger of computing policies that will make the system move in circles and never reach the goal. For instance, if we minimize the expected cost and use a discount factor of 0.9 it takes the system on average 18,000 moves to reach the goal and the average cost is 38,755. Is is intuitive that $\gamma = 0.9$ is not the proper choice for the discount factor.

Go back to Step 1

**Step 1:** Pick $\gamma = 0.995$.

**Step 2:** We solve the exact DP using $\lambda = 1$ .

- The produced policy leads the system to the goal in a finite number of steps.

**Step 3:** Decrease the risk averse parameter $\lambda_1$ by 0.01 until $\lambda_1 = 0$ and solve the exact DP.

- The range of $\lambda_1$ for which we can recover valid policies is $\lambda_1 = [0.91, 1]$.

**Step 4:** By testing these policies via Monte Carlo simulations, we realize that, up to this point, we have created 2 policies that belong to the multistage mean-CVaR efficient frontier. The most risk averse policy is the one, which corresponds to the least multistage $CVaR_\eta$. This policy is produced when $\lambda_1 = 0.99$.

According to our methodology one should go to step 1 and increase the discount factor $\gamma$ and simultaneously at step 3 and manipulate the increment 0.01. Here, we left $\gamma$ unchanged but we do manipulate the increment of change from 0.01 to 0.0001. For $\lambda_1 = 0.9993$, we retrieve the policy with the characteristics as shown in Table 26.

The cost distribution produced when each policy is applied to 1,000 independent scenarios is shown next.

**Table 26:** For the multi stage problem ($\mathbf{p} = 0.8$) this table demonstrates: the policies derived from the mean-CVaR tradeoff. For each policy we demonstrate its mean $\mu$ and the evaluation of the risk measures ($\sigma$,$CVaR_{0.95}$ ) associated with it.

| Risk Averse Parameter | $\mu$ | Standard Deviation | $CVaR_{0.95}$ | Expected Number Of Steps To Reach Goal |
|---|---|---|---|---|
| $\lambda_1 = [1]$ | 194 | 757 | 1,909 | 82.5 |
| $\lambda_1$= [0.9993] | 218 | 505 | 1,506 | 78.8 |
| $\lambda_1$= [0.91-0.99] | 239 | 360 | 999 $\pm$ 19 | 78.8 |

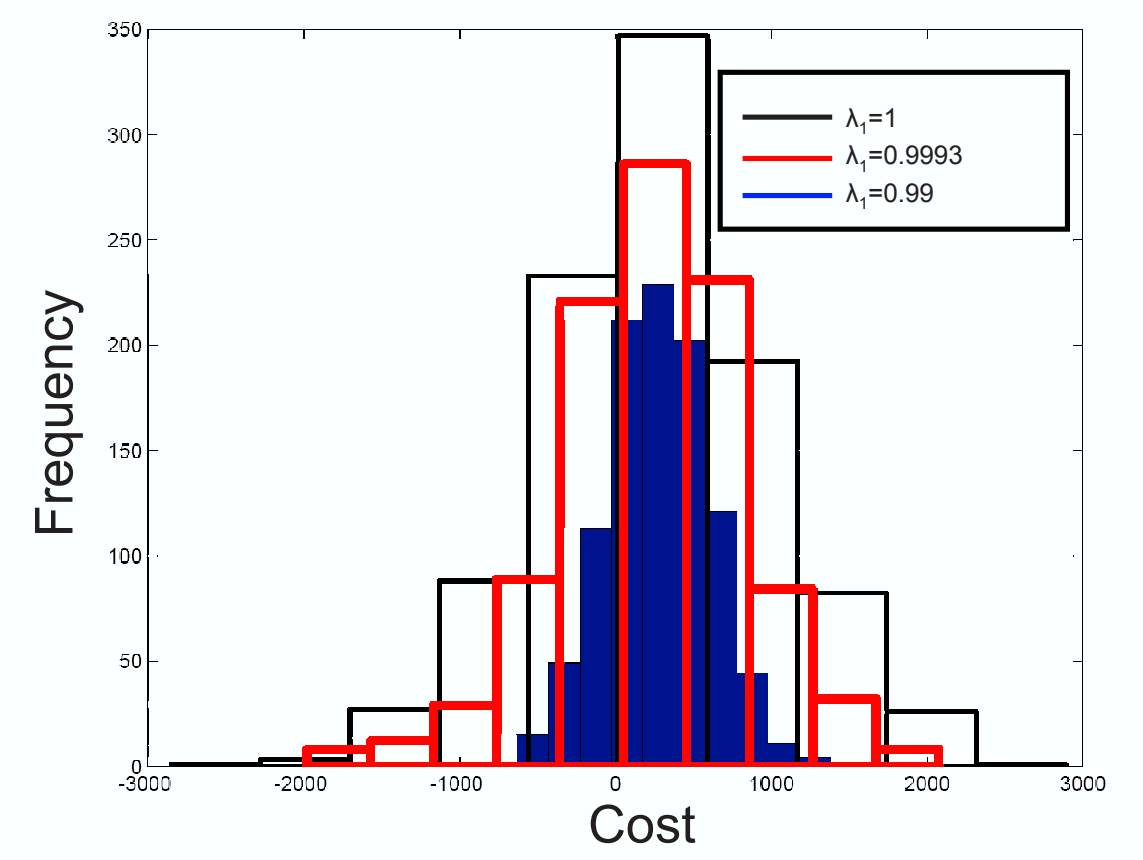



**Figure 49:** The cost distribution produced for the 900 discrete state space stochastic shortest path problem with probabilistic transitions, when we apply the policies generated by the parameters as instructed by Table 26.

## *6.8 Off-line Approximate Dynamic Programming Applied To The 900 Discrete State Stochastic Shortest Path Example*

In this section, we will use the single period mean-$CVaR_{0.95}$ utility within the ADP approach as delineated in the previous chapter.

First, we set the discount factor $\gamma = 0.995$ and choose $\lambda_1 = 1$, $\lambda_1 = 0.8$ and $\lambda_1 = 0.2$, then we transform the single stage cost function via $f_t(s_t, \lambda_1) = \lambda_1 \mu(s_t) + (1 - \lambda_1) CVaR_{0.95}(f(s_t))$ and solve the corresponding deterministic LP's. The LP solutions are specific state trajectory that drive the system from the starting to the goal state. These trajectories are shown in Fig.50 For $\lambda_1 = \{1, 0.8, 0.2\}$, after applying the off-line ADP

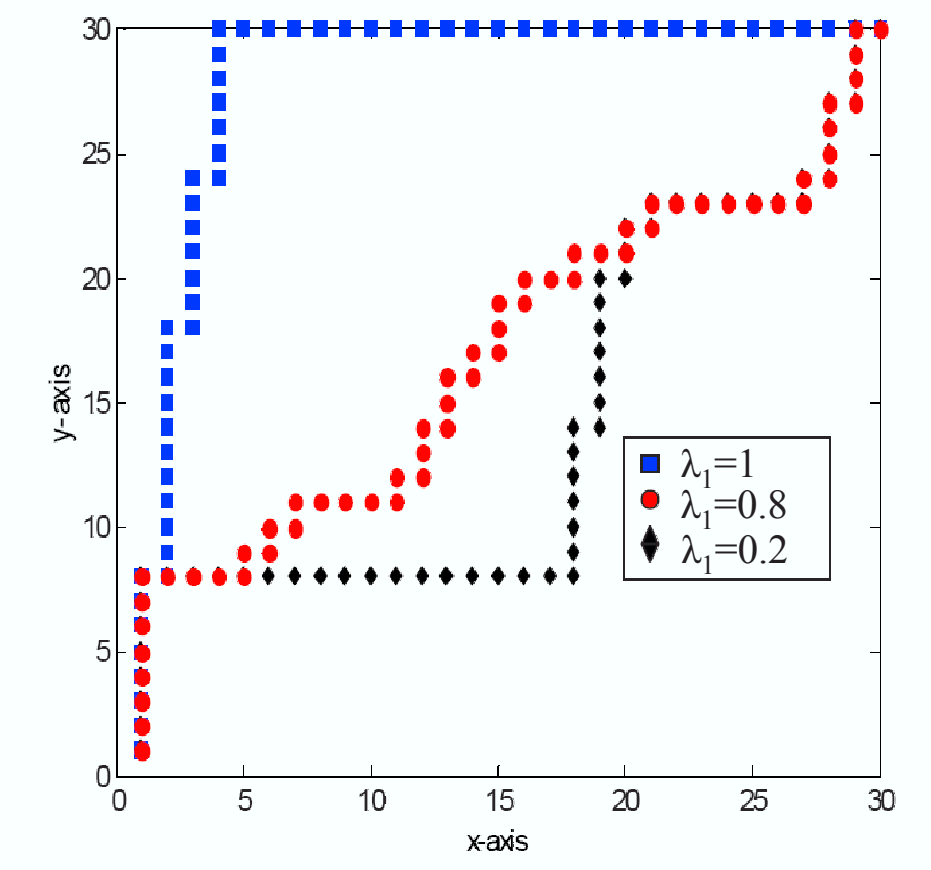



**Figure 50:** Initialization of the value table using deterministic optimization. We call the LP optimization routine after transforming each states non positive profit using $\lambda\mu$+(1-$\lambda$)CVaR.

algorithm, we end up collecting for each $\lambda_1$ the state space depicted in Fig.51.

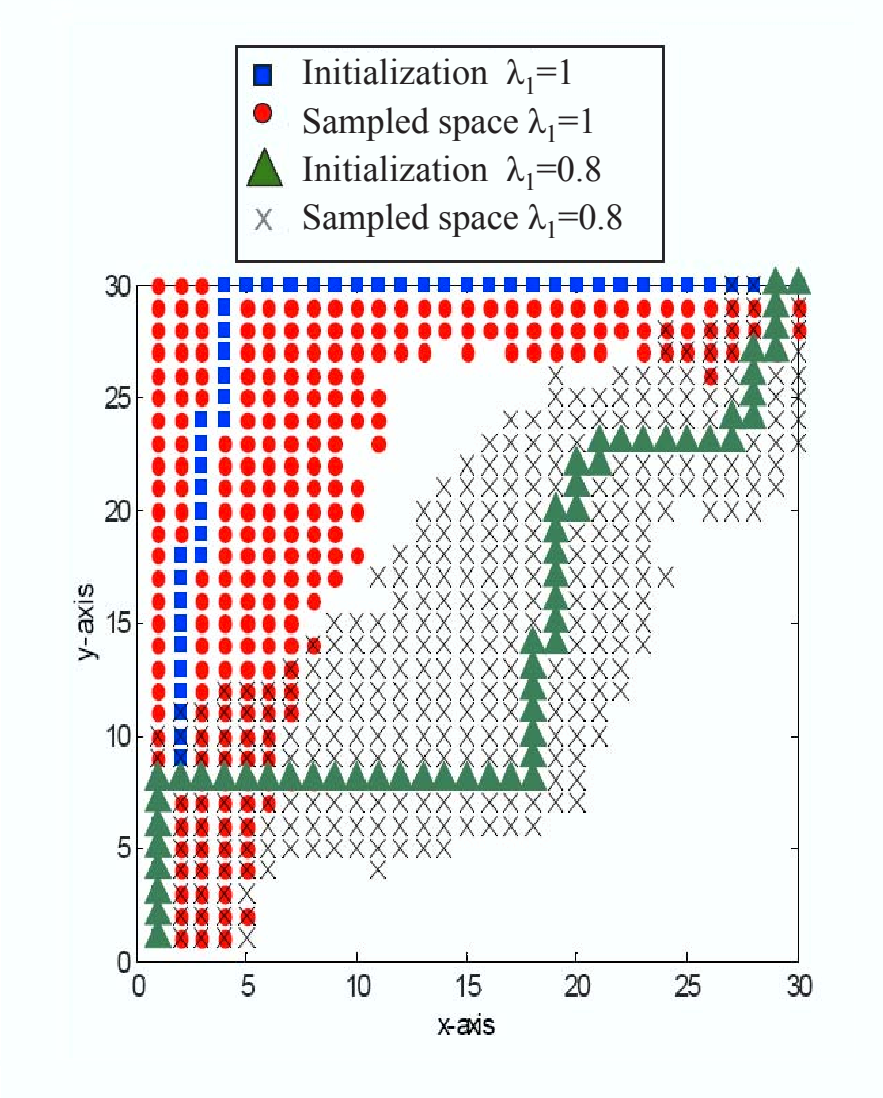



**Figure 51:** Sampled state space after applying the Offline-ADP routine.

The derived policies are evaluated with respect to 1,000 MC simulations. These results represent pareto multistage mean-CVaR solutions. Nonetheless, there is no guarantee that this methodology will always provide such solutions. In order for this procedure to have a high chance to provide such solutions we would need: a) the initialization procedure to be able to provide radically different state trajectories leading the system from the starting state to the goal state, b) these state trajectories should have neighboring states with similar characteristics mainly with respect to the single stage profit/cost and variance.

**Table 27:** Cumulative results using the ADP strategy with the proposed summation of single stage mean-CVaR functions on a 900 discrete state instance when $\lambda_1 = 1$ and $\lambda_1 = 0.8$ .

| Risk Averse Parameter Vs Probability Of Successful Transition | $\mathbf{p} = 0.8$ |
|---|---|
| $\lambda_1 = 1$ | $\mu = 259$<br>$\sigma = 452$<br>$CVaR_{0.05} = 1,216$ |
| $\lambda_1 = 0.8$ | $\mu = 308.1$<br>$\sigma = 316.16$<br>$CVaR_{0.05} = 991$ |

## 6.9 Chapter Conclusions

In this chapter, we initially summarized the ways that one can induce risk-time preferences in MDP's. We proposed a linear single period utility as a mean to express such preferences.

The discount factor controls the horizon effect, while the myopic risk averse parameter determines if we will capture the multistage risk effect or not. By setting the discount factor, the basis to identify multistage risk averse policies instructs to weigh with a parameter $\lambda_1$ the single stage expected performance and by a $1-\lambda_1$ the single stage downside risk measured by $CVaR_\eta$. The understanding of the ratio of these myopic weights is the key to produce a multi stage efficient frontier.

Numerically and analytical examples demonstrate that when we utilize discount factors closer to 1 then only by varying this risk averse parameter within a small range close to 1 we get create pareto solutions consistent with the multistage mean-CVaR efficient frontier.

The major contribution of this chapter is the understanding of the weighting of the proposed myopic utility that if used within DP as well as ADP approaches will generate a good approximation of the exact multistage mean-CVaR efficient frontier.

# CHAPTER 7

# CONCLUSION AND FUTURE WORK

In this chapter I summarize this work and present some possible extensions for future work.

## *7.1 Summary*

The scientific domain of this thesis is optimization under uncertainty and risk sensitive objectives for discrete event stochastic systems under . In particular, this thesis focuses on the practical implementation of the Dynamic Programming (DP) methodology to discrete event stochastic systems.

Specifically, for the purposes of this thesis we developed the following ADP techniques. The first one is inspired from the Reinforcement Learning (RL) literature and is termed as Real Time Approximate Dynamic Programming (RTADP). The RTADP algorithm is meant for active learning while operating the stochastic system. The basic idea is that the agent while constantly interacts with the uncertain environment accumulates experience, which enables him to react more optimal in future similar situations. While the second one is an off-line ADP procedure. Both approaches are developed for discrete event stochastic systems and their main focus is the controlled exploration of the state space circumventing in such a way one of the severe computational obstacles of DP that is related with the cardinality of the state space.

These ADP techniques are demonstrated on a variety of discrete event stochastic systems such as: **i)** a three stage queuing manufacturing network with recycle, **ii)** a supply chain of the light aromatics of a typical refinery and **iii)** several stochastic shortest path instances with a single starting and terminal state.

Moreover, this work addresses, in a systematic way, the issue of approximating the exact multistage mean-CVaR tradeoff with the summation of a proposed intra-period mean-CVaR utility. We identify the source of the deviation between these criteria via a small analytic

example , which will provide to us the intuition to complete the structure of the proposed linear intra-period utility The proposed structure of the linear single stage intra-period utility is composed out of a linear combination of the two presumed single stage statistics with corresponding weights. This thesis opens the research question of a systematic adaptation of these weights to yield a pareto efficient frontier.

A really significant contribution of this work is that this proposed utility can be used along every ADP or exact DP to come up with pareto efficient policies. In fact we briefly attempted to intergrade the developed offline-ADP procedure with the proposed approach and we have yielded at least for on instance ADP risk sensitive policies (Section 6.6).

## *7.2 Future Work: Short Term*

This thesis developed empirical ADP algorithms for risk-sensitive planning. The treatment is, however, far from complete. Here, we list some possibilities for future research.

### 7.2.1 Using Aggregation In RTADP Algorithm

The first iteration of chapter 3 was written using the concept of aggregation around macro-states [96]. The state space was assumed to be composed out of finite macro-states. The macro-state is composed out of many system states as defined by some physical criteria. The difficulty lies in the transition equation for macro-states, since for each action we would need to simulate the transition from a system state within the macro-state to the corresponding successive state and then retrieve its macro-state. An interesting research would be to develop a mechanism that would automatically conclude to a right macro-state structure without affecting the quality of the learned policy.

### 7.2.2 Guided State Space Exploration In A Multi-stage Setting Via A Lower And Upper Bounding Mathematical Programming Scheme

In the proposed ADP methodologies, the state space exploration was performed via: a) $\epsilon$-greedy actions or b) MC simulations with the so far optimal policy . A quantitative way to guide the exploration of the state space would be via the usage of an upper and lower bounding mathematical programming scheme. For a cost setting, one can use sample

average approximation to generate a numerical upper bound, while one can use perfect information (as done in Chapter 4) to generate a lower bound. There may be theoretical worries whether these two methodologies indeed yield valid upper and lower bounds in a multi-stage setting. After gathering this sort of information we can use it as a prior to introduce bias with where we would want to explore or not, similar to the heuristic bias introduced by [17, 18].

### 7.2.3 To Device A Procedure That Automatically Tunes The Weights Of The Linear Mean-CVaR Objective Function Within DP To Well Approximate Multi-stage Mean-CVaR Trade Off

In chapter 6, we identified the source of the deviation between the initially thought appropriate linear intra-period utility to approximate the exact multi-stage Mean-CVaR. Specifically, we introduced a hyper-parameter to complete its structure.

A complete understanding, about the connection between the discount factor as well as the primary risk averse parameter choice with respect to the selection of proper weighting the proposed linear objective, is needed. We emphasize again that given the proper weighting, this linear utility can be used with exact DP as well as within all the constructed ADP schemes and derive a wide spectrum of pareto efficient policies.

### 7.2.4 Simulation Results On A Large Scale Project Portfolio Problem Using The Off-line Risk Sensitive ADP Methodology

The proposed tools can be used for a wide range of problem that

Extensive simulations for the problem as presented as chapter 7 are pending. We need to propose a proper weighting scheme for the linear mean-CVaR utility, in order to generate a wide spectrum of policies for this application. In order make the problem more realistic and more challenging we will relax the resource constraint to enlarge the state space and use ADP along with the proposed weighted linear risk averse utility.

## *7.3 Future Work: Long Term*

Currently, the foundation of the Process System Engineering community is based on mathematical programming. The main reasons are the reliable algebraic computer languages,

(e.g. GAMS , G-PROMS, etc.) which allow the compact representation of the constraints, where one line of computer code can generate thousands of constraints and the superb solvers that solve MILP formulations very efficiently (CPLEX V.11).

Nonetheless, a different way to solve for stochastic application is via ADP methods. Undoubtedly, these DP based methods treat endogenous uncertainty in a more natural way. Currently we have empirical ADP approaches that work better than rolling horizon MILP's. For instance for large scale resource allocation problems, utilizing ADP approaches the numerical exercises of Powell [1] prove that fact. But still its not proven that they have an advantage over rolling horizon stochastic optimization methods.

In order for the PSE community to be susceptible to ADP methodologies, one should develop a tool similar to GAMS and a modelling language that will make ADP attractive and easy to use. I would characterize this as my abmitious long term project.

## .1 Linear Programming Formulation

Assume the shortest path is to be determined between node 1 and $n$, where $n$ is the total number of network nodes. A single branch connects node $i$ with node $j$. It is assumed that a single branch connects $i$ and $j$ and that we do not consider flows in the opposite direction. Thus we wish to find the set of $x_{i,j}$ to

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} f_{i,j} x_{i,j} \\ s.t. \quad & \sum_{k=2}^{n} x_{1,k} = 1 \\ & \sum_{k=1}^{n-1} x_{k,n} = 1 \\ & \sum_{i=1}^{n} x_{i,k} - \sum_{j=1}^{n} x_{k,j} = 0 \quad \forall k = 2, ...n-1 \\ & x_{i,j} \in 0, 1 \quad \forall i, j \end{aligned} \tag{91}$$

, where $f_{i,j}$ is the cost from node $i$ to node $j$.

The first two constraints assure that one action is taken from the first node, and also one action will lead to the final node. The first constraint forces the actions into a node to be equal to the ones directed out of the node. This formulation even if $x_{i,j}$ are binaries is an LP, since the corresponding coeffient matrix is unimodular.

## .2 Solving Large Shortest Path Instances Via Systematic Initialization

Solving shortest path instances with goal states that enumerate more than 10,000 states is a computational prohibitive task. In this case, we would need to solve the long run average problem ($\gamma = 1$) and not a discounted horizon problem. Otherwise the solution from the discounted horizon problem is very likely to force the system into cycles and never to visit the goal To address such large instances using the off-line ADP approach, we device a generic initialization procedure.

### .2.1 Initialization Procedure

This procedure decomposes the problem to small independent problems providing the opportunity for parallel computing.

The primary goal given this procedure is to create a partial policy considering only a small portion of the state space. This policy will drive the system from the start to the goal state with probability approaching one. In our numerical experiments the goal state is a singleton.

**Subroutine Name: Initialization - Shortest Path Problem**

**Create A Path from $s_0$ to $s_G$** . Run deterministic optimization and compose a set that is defined via the following path of states

$\{s_0, s_1, ..., s_n, ..., s_{2n}, ..., s_{3n}, ..., s_{(\kappa-1)n}, ..., s_G\}$

**Create $\kappa$ Subproblems**

$1^{st}$ subproblem is defined by: Starting state $s_0$. Goal State $s_n$.

$2^{nd}$ subproblem is defined by : Starting state $s_n$. Goal State $s_{2n}$.

$\kappa^{th}$ subproblem : Starting state $s_{(\kappa-1)n}$. Goal State $s_G$.

**Use the off-line ADP Approach as delineated in Chapter 5 For Each Subproblem**

Set $\Theta$=0 for each subproblem.

The output of subproblem $i$ is a set of sampled stated denoted as $M_i$

**Initialize State Space $\mathbb{S}_0$**

$\mathbb{S}_0 = \cup_i M_i$

**Initialize The Value Functions $\forall s_t \in \mathbb{S}_0$**

Starting from the $k^{th}$ subproblem and going backwards.

The value function estimations of the states sampled by the $\kappa^{th}$ subproblem remain the same. The value function estimations of the states sampled by the $i^{th}$ subproblem are

updated as follows: we add to all the state estimations the value function estimate of the starting state of the $i^{th} + 1$ subproblem.

# Bibliography


[1] Powell, W. B., *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley and Sons, 2007.

[2] Bellman, R. E., *Dynamic Programming*. New Jersey: Princeton University Press, 1957.

[3] Liptak, B. G., *Instrument Engineers' Handbook: Process Control*. CRC Press, 1995.

[4] Lee, J., "Robust inferential control : A methodology for control structure selection and inferential control system design in the presence of model/plant mismatch.," *http://resolver.caltech.edu/CaltechETD:etd-11162005-134952*, 1991.

[5] Elberly, J. C. and Mieghen, J. A., "Multi-factor dynamic investment under uncertainty," *Journal of Economic Theory*, vol. 75, pp. 345–387, 1997.

[6] Santoso, T. Ahmed, S. G. M. and Shapiro, A., "A stochastic programming approach for supply chain network design under uncertainty," *European Journal of Operational Research*, pp. 96–115, 2005.

[7] Rogers, M.J. Gupta, A. and Maranas, C., "Real options based analysis of optimal pharmaceutical research and development portfolios," *Ind. Eng. Chem. Res.*, vol. 41, pp. 6607–6620, 2002.

[8] Mudchanatongsuk, S. Primbs, J. and Wong, W., "Optimal pairs trading: A stochastic control approach," *Submitted to ACC*.

[9] Primbs, J., "Dynamic hedging of basket options under proportional transaction costs using receding horizon control," *Submitted. Also availabe at http://www.stanford.edu/ japrimbs/AutoSubmit20070813.pdf*.

[10] Ulu, C. Smith, J., "Information acquisition and technology adoption," *Technology Management for the Global Future*, vol. 4, pp. 1693–1693, 2006.

[11] Puterman, M. L., *Markov Decision Processes*. New York, NY: Wiley, 1994.

[12] Chung, K.-J. and Sobel, M. J., "Discounted mdps: Distribution functions and exponential utility maximization.," *SIAM Journal of Control and Optimization*, vol. 35, no. 1, pp. 49–62, 1987.

[13] Sobel, M., "Mean-variance tradeoffs in an undiscounted mdp," *Operations Research*, vol. 42, no. 1, pp. 175–183, 1994.

[14] Liu, Y., *Decision-Theoretic Planning Under Risk-Sensitive Planning Objectives*. PhD Thesis, Georgia Insitute Of Technlogy, 2003.

[15] Barto, A. Bradtke, S. and Singh, S., "Learning to act using real-time dynamic programming," *Artificial Intelligence*, vol. 72, pp. 81–138, 1995.

[16] Sutton, R. S. and Barto, A. G., *Reinforcement Learning: An Introduction*. A Bradford Book. MIT Press, Cambridge, MA, 1998.

[17] Choi, J., Realff, M. J., and Lee, J. H., "An algorithmic framework for improving heuristic solutions: Part i. a deterministic discount coupon traveling salesman problem," *Computers and Chemical Engineering*, vol. 28, no. 8, pp. 1285–1296, 2004.

[18] Choi, J., Lee, J. H., and Realff, M. J., "An algorithmic framework for improving heuristic solutions: Part ii. a new version of the stochastic traveling salesman problem," *Computers and Chemical Engineering*, vol. 28, no. 8, pp. 1297–1307, 2004.

[19] *http://stoprog.org/index.html/spintroduction.html*.

[20] Huang, K. and Ahmed, S., "The value of multi-stage stochastic programming in capacity planning under uncertainty," *To appear in Operations Research*, 2008.

[21] Verweij, B., Ahmed, S., Kleywegt, A., G., N., and A., S., "The sample average approximation method applied to stochastic routing problems: A computational study," *European Journal of Operational Research*, vol. 24, no. 2, pp. 289–333, 2003.

[22] Mendez, C., Cerda, J., Grossmann, I., Harjunkoski, I., and Fahl, M., "State-of-the-art review of optimization methods for short-term scheduling of batch processes," *Computers and Chemical Engineering*, vol. 30, pp. 913–946, 2006.

[23] Dasika, M., Gupta, A., and Maranas, C., "A mixed integer linear programming framework (milp) for inferring time delay in gene regulatory networks," *Pacific Symposium on Biocomputing*, 2004.

[24] Lee, H., Pinto, J., Grossmann, I., and Park, S., "Milp model for refinery short term scheduling of crude oil unloading with inventory management," *Industrial and Engineering Chemistry Research*, vol. 35, no. 5, pp. 1630–1641, 1994.

[25] Pinto, J. and Grossmann, I., "A continuous time milp model for short term scheduling of batch plants with preordering constraints," *Computers and Chemical Engineering*, vol. 20, pp. 1197–1202, 1996.

[26] Blomvall, J. and Shapiro, A., "Solving multistage asset investment problems by the sample average approximation method.," *Mathematical Programming: Series A and B*.

[27] Pinto, J. and Moro, G., "Planning and scheduling for refinery operations," *Computers And Chemical Engineering*, vol. 24, pp. 2259–2276, 2000.

[28] Stokey, N. and Lucas, R.E. with Prescott, E., *Recursive Methods in Economics Dynamics*. Harvard University Press, 1989.

[29] Bertsekas, D. P., *Dynamic Programming and Optimal Control*, vol. II. Athena Scientific, 1995.

[30] Thrun, S. and Schwartz, A., *Issues in using function approximation for reinforcement learning*. Proceedings of the Fourth Connectionist Models Summer School (Hillsdale, NJ) Lawrence Erlbaum,, 1993.

[31] Cervellera, C., Chen, V. C. P., and Wen, A., "Optimization of a large-scale water reservoir network by stochastic dynamic programming with efficient state space discretization," *European Journal of Operational Research*, vol. 171, pp. 1139–1151, 2006.

[32] Tsai, J. C. C., Chen, V. C. P., Beck, M. B., and Chen, J., "Stochastic dynamic programming formulation for a wastewater treatment decision-making framework. annals of operations research," *Special Issue on Applied Optimization Under Uncertainty*, vol. 13, 2004.

[33] Lee, J. M., Kaisare, N., and Lee, J. H., "Choice of approximator and design of penalty function for an approximate dynamic programming based control approach," *Journal of Process Control*, vol. 16, pp. 135–156, 2006.

[34] De Farias, D. and Van Roy, B., "The programming approach to approximate dynamic programming," *Operations Research*, vol. 51, no. 6, pp. 850–865, 2003.

[35] Powell, W., A. George, B. B.-A., and Simao, H., "Approximate dynamic programming for high dimensional resource allocation problems," *Proceedings of the IJCNN, Montreal*, 2005.

[36] Powell, W. B. and Van Roy, B., "Approximate dynamic programming for high dimensional resource allocation problems," *Proceedings of the IJCNN, Montreal*, 2005.

[37] MacQueen, J., "A modified dynamic programming method for markovian decision problems," *Journal Mathematical Analytical Applications*, vol. 14, pp. 38–43, 1966.

[38] Chang, H., Fu, M., Hu, J., and Marcus, S., *Simulation-Based Algorithms for Markov Decision Processes*. Berlin: Springer, 2007.

[39] Trevor, H., Tibshirami, R., and Friedman, J., *The Elements of Statistical Learning*. Springer, 2001.

[40] Artzner, P., Delbaen, F., Eber, J.-M., and Heath, D., "Coherent measures of risk," *Mathematical Finance*, vol. 9, pp. 203–228, 1999.

[41] Benfield, *Risk Measurement in Insurance*. London: http://www.benfieldgroup.com/media+centre/research+and+publications/risk+measurement.pdf, 2005.

[42] Markowitz, H., "Portfolio selection," *Journal of Finance*, pp. 77–91, 1952.

[43] Neumann, J. and Morgenstern, O., *Theory of Games and Economic Behavior*. Princeton, NJ. Princeton University Press, 1944.

[44] Koopman, B., "The axioms and algebra of intuitive probability," *Annals of Mathematics*, vol. 41, pp. 269–292, 1940.

[45] Koopman, B., "The basis of probability," *Bulletin of the American Mathematical Society*, vol. 46, pp. 763–774, 1940.

[46] Morse, P., "In memoriam: Bernard osgood koopman, 1900-1981," *Operations Research*, vol. 30, no. 3, pp. 417–427, 1982.

[47] Avila-Godoy, M., "Controlled markov chains with exponential risk-sensitive criteria: Modularity, structured policies and applications.," *PhD Thesis, University Of Arizona*.

[48] Filar, J., Kallenberg, L., and Lee, H., "Variance penalized markov decision processes," *Mathematics Of Opearations Research*, vol. 14, no. 1, pp. 147–161, 1989.

[49] Markowitz, H., *Mean Variance Analysis in Portfolio Choice and Capital Markets*. Oxford: Blackwell, 1987.

[50] Rockafellar, R. and Uryasef, S., "Optimization of conditional value-at-risk," *The Journal of Risk*, vol. 2, pp. 21–41, 2000.

[51] Borca, B., *Multi-period Constrained Portfolio Optimization Using Conditional Value at Risk*. University of Lausanne: Master of Sciencein Banking and Finance, 2004-2005.

[52] Cheng, L., SUbrahmanian, E., and Westerberg, A., "A comparison of optimal control and stochastic programming from a formulationa and computation perspective," *Computers and Chemical Engineering*, vol. 29, pp. 149–164, 2004.

[53] Cheng, L., SUbrahmanian, E., and Westerberg, A., "Design and planning under uncertainty: Issues on problem formulation and solution," *Computers and Chemical Engineering*, vol. 27, pp. 781–801, 2003.

[54] Cheng, L., SUbrahmanian, E., and Westerberg, A., "Multi-objective decisions on capacity planning and inventory control," *Industrial and Engineering Chemistry Research*, vol. 43, pp. 2192–2208, 2004.

[55] Li, D., "Multiple objectives and non-separability in stochastic dynamic programming," *International Journal of Systems Science*, pp. 933–850, 1990.

[56] Liu, L. and Koenig, S., "An exact algorithm for solving mdps under risk-sensitve planning objectives with one-switch utility functions.," *In Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[57] Jung, J. Y., Blau, G., Pekny, J. F., Reklaitis, G. V., and D., E., "A simulation based optimization approach to supply chain management under demand uncertainty," *Computers and Chemical Engineering*, vol. 28, pp. 2087–2106, 2004.

[58] Subramanian, D., Pekny, J., and Reklaitis, G. V., "Simulation-optimization framework for stochastic optimization of research and development pipeline management," *AICHE Journal*, vol. 1, pp. 96–112, 2003.

[59] Wan, X., Pekny, J., and G.V.Reklaitis, "Simulation based optimization for risk management in multi-stage capacity expansion," *Computers and Chemical Engineering*, vol. 28, pp. 971–983, 2004.

[60] Jung, J. Y., Blau, G., Pekny, J. F., Reklaitis, G. V., and Eversdyk, D., "A simulation based optimization approach to supply chain management under demand uncertainty," *Computers and Chemical Engineering*, vol. 28, no. 10, pp. 2087–2106, 2004.

[61] Wong, W. C. and Lee, J. H., "Disturbance modeling for process control via hidden markov models," *Dycopts Conference*, 2007.

[62] Gullapalli, V. and Barto, A., "Convergence of indirect adaptive asynchronous value iteration algorithms," *Advances in Neural Information Processing Systems*, vol. 6, pp. 695–702, 1994.

[63] Barto, A., Bradtke, S., and Singh, S., "Learning to act using real time dynamic programming," *Artificial Intelligence*, vol. 72, pp. 81–138, 1995.

[64] Lee, J. M. and Lee, J. H., "Approximate dynamic programming strategies and their applicability for process control: A review and future directions," *International Journal of Control Automation and Systems*, vol. 2, no. 3, pp. 263–278, 2004.

[65] Dixit, A. and Pindyck, R. S., *Investment Under Uncertainty*. Princeton University Press, Princeton, New Jersey, 1993.

[66] Mieghem, J. A. V., "Capacity management investmentand hedging: Review and recent developments," *Manufacturing and Service Operations Management*, vol. 5, no. 4, p. 269 302, 2003.

[67] Eberly, J. C. and Van Mieghem, J. A., "Multi-factor dynamic investment under uncertainty," *Journal of Economic Theory*, vol. 75, no. 2, pp. 345–387, 1997.

[68] Birge, J. and Louveaux, F., *Introduction to Stochastic Programming*. Springer and Verlag, 1997.

[69] Shapiro, N., "An adaptive sapling algorithm for solving markov decision processes," *Mathematical Methods of Operations Research*, vol. 53, no. 1, pp. 126–68, 2005.

[70] Brafman, R. and Tennenholtz, M., "R-max-a general polynomial time algorithm for near optimal reinforcement learning," *Journal of Machine Learning Research*, vol. 3, pp. 213–231, 2002.

[71] Kearns, Michael. Mansour, Y. and Ng, A. Y., "A sparse sampling algorithm for near-optimal planning in large markov decision processes," *IJCAI*, pp. 1324–1231, 1999.

[72] Yi, G. and Reklaitis, G., "Optimal design of batch-storage network considering exchange rates and taxes," *AIChE Journal*, vol. 53, pp. 1211–1235, 2007.

[73] Yi, G. and Reklaitis, G., "Optimal design of batch-storage network with uncertainty and waste treatment," *AIChE Journal*, vol. 52, pp. 3473–3490, 2006.

[74] Kuo, T. and Chang, C., "Optimal planning strategy for the supply chains of light aromatic compounds in refineries," *Computers and Chemical Engineering*, vol. 32, pp. 1147–1166, 2008.

[75] McManan, B., Likhachev, M., and Gordon, G. J., "Bounded real-time dynamic programming: Rtdp with monotone upper bounds and performance guarantees," *Appearing in Proceedings of the 22 International Confer- ence on Machine Learning, Bonn, Germany*, 2005.

[76] Hegazy, T., *Distributed Approach to Dynamic Autonomous Agent Placement for Tracking Moving Targets*. PhD Thesis, Georgia Institute Of Technology, 2004.

[77] Nair, R. Pynadath, D.-Y. M. T. M. and Marsella, S., "Taming decentralized pomdps: Towards efficient policy computation for multiagent settings.," *In Proceedings of International Joint Conference on Artificial Intelligence*, 2003.

[78] Pynadath, D. and Tambe, M., "The communicative multiagent team decision problem: Analyzing teamwork theories and models.," *In Journal of Artificial Intelligence Research,*, 2002.

[79] Peshkin, L. Kim, K.-E. M. N. and Kaelbling, L. P., "Learning to cooperate via policy search.," *In Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2000.

[80] Balasubramanian, J. and Grossmann, I. E., "Approximation to multistage stochastic optimization in multiperiod batch plant scheduling under demand uncertainty," *Ind. Eng. Chem. Res.*, vol. 42, pp. 3695 –3713, 2004.

[81] Balasubramanian, J. and Grossmann, I. E., "Scheduling optimization under uncertainty - an alternative approach," *Computers and chemical engineering*, vol. 27, pp. 469–490.

[82] Diaz-Banez, M., Gomez, F., Toussaint, G., Ramezani, V., and Marcus, S. I., "Computing shortest paths for transportation of hazardous materials in continuous spaces," *J. Food Eng.*, vol. 70, no. 3, pp. 293–298, 2004.

[83] Nannicini, G. Baptiste, P. B. G. K. D. and Liberti, L., "Fast paths in large-scale dynamic road networks," *Computational Optimization and Applications*, p. Published online, 2008.

[84] Endelman, J., Gomez, F., and Toussaint, G., "Site-directed protein recombination as a shortest path problem.," *Prot. Eng. Design and Selection*, vol. 17, no. 7, pp. 589–594, 2004.

[85] Dijkstra, E., "A note on two problems in connexion with graphs," *Numerische Mathematik*, p. 269271, 1959.

[86] Russell, S. J. and Norvig, P., *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[87] Si, J., Barto, A., Powell, W., and Wunsch, D. e., *Learning and Approximate Dynamic Programming: Scaling up to the Real World*. New York: John Wiley and Sons, 2004.

[88] Pratikakis, N., Realff, M. J., and Lee, J. H., "Strategic capacity decisions in manufacturing using stochastic dynamic programming," *Naval Research Logistics*, Under Review.

[89] Smith, T. and Simmons, R. G., "Focused real-time dynamic programming for MDPs: Squeezing more out of a heuristic," in *Proc. Nat. Conf. on Artificial Intelligence (AAAI)*, 2006.

[90] Pratikakis, N., Lee, J. H., and Realff, M. J., "A real time adaptive dynamic programming approach for planning and scheduling," *$16^{th}$ ESCAPE - $9^{th}$ PSE*, pp. 1179–1185.

[91] Sobel, M., "Discounting and risk neutrality," *Technical Memorandum Number 724F*.

[92] Corner, J. and Corver, P., "Characteristics of decisions in decision analysis practice.," *The Journal of Operational Research Society*, vol. 46, no. 1, pp. 304–314, 1995.

[93] Howard, R. A. and Matheson, J. E., "Risk-sensitive markov decision processes.," *Management Science*, vol. 18, no. 7, pp. 356–369, 1972.

[94] Jaquette, S., "Markov decision processes with a new optimality criterion: Discrete time.," *The Annals of Statistics*, vol. 1, no. 3, pp. 496–505, 1973.

[95] Corallupi, S. and Marcus, S. I., "Risk-sensitive control of markov decision processes.," *In Proceedings CISS*.

[96] Bertsekas, D. and Castanon, D., "Adaptive aggregation methods for infinite horizon dynamic programming," *IEEE Transactions on Automatic Control*, no. 6, pp. 589–598, 1989.